
MAGAZINE MSX
AÑO I
Núm. 5
Septiembre
1985
250 Ptas.
Comandos E/S: Cómo entenderse con el ordenador
Diseña tus propios sprites
Código máquina ese desconocido
El BASIC MSX comparado con Spectrum y Commodore 64
COLOR
BLOAD
LOCATE
SPRITE$
ANDRÉS SÁNCHEZ

¡EL IMPERIO CONTRAATACA!













Ordenador Personal
TOSHIBA HX-10
Su Ordenado Servidor

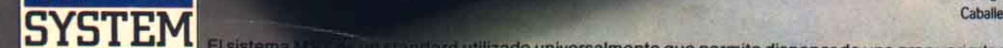
69.500 Ptas.



Características principales:
Sistema standard MSX. Memoria de 64 K RAM, 32 K ROM y 16 K de pantalla. 16 colores. 73 teclas. 32 sprites. Sistema multicolor: 64 x 48 bloques. Sonido: 8 octavas tres acordes. Conexiones para: casette, impresora, 2 mandos y futuras expansiones.

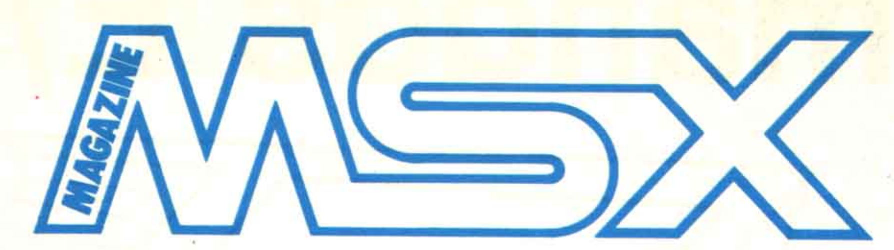


TOSHIBA
española de microordenadores s.a.
Caballero, 79 - Tel. 321 02 12 - Telex 97087 EMOS - 08014 BARCELONA



El sistema MSX es un standard utilizado universalmente que permite disponer de una gran variedad de programas y accesorios compatibles entre sí.

PUZZLE



AÑO I - Núm. 5
Septiembre 1985

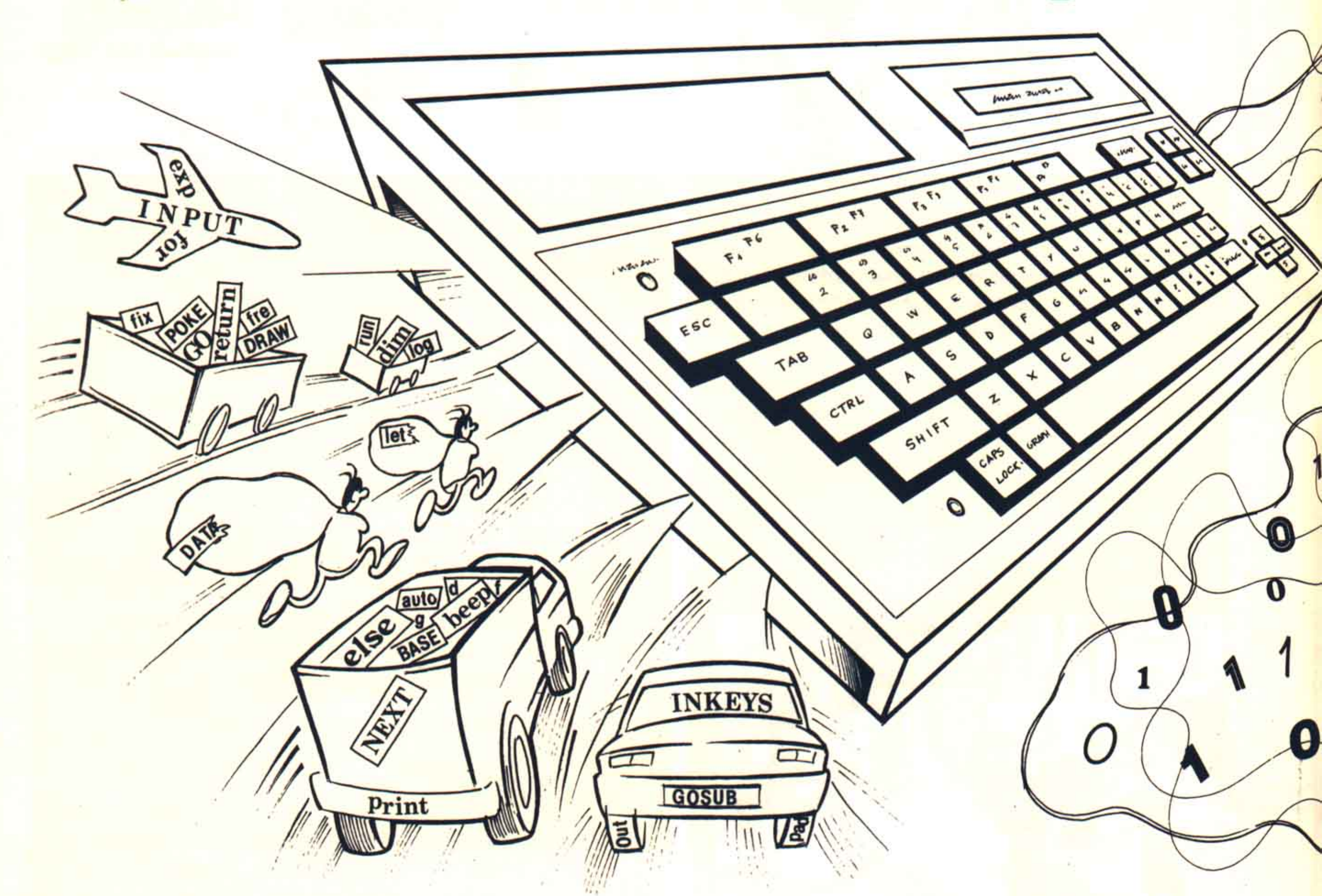
Parece que fue ayer cuando empezaron las vacaciones. Ahora el verano se acaba, y con él se inicia una nueva etapa, la del curso escolar. Las noticias, algo paralizadas desde el comienzo de la época estival, vuelven a surgir; nuevos juegos, libros, etc., prometen afianzar un poco más el estándar MSX en un mercado en que el ordenador personal se está convirtiendo en máquina indispensable a cualquier nivel.

A puertas de importantes ferias de informática, el estándar sigue sumando puntos.





Comandos E/S e



Dentro de los diferentes comandos que presenta el BASIC MSX se pueden hacer varios bloques dependiendo de la función general que desempeñan, así por ejemplo podemos distinguir comandos de cálculo como pueden ser todas las funciones matemáticas, funciones lógicas y demás, comandos de comunicación del ordenador con el ámbito exterior, es decir los periféricos, que podemos agrupar genéricamente como comandos entrada-salida o abreviadamente comandos E/S, etc.

En este artículo nos vamos a centrar en este último tipo, es decir los comandos E/S.

Agrupar los comandos BASIC en un solo bloque, aparte de ser algo pretencioso, sólo nos puede servir de base para tratarlos más profundamente y así, a la hora de estudiarlos debemos subdividirlos en varios apartados si no queremos caer en la trampa de la superficialidad.

En un principio, los comandos a tratar los vamos a agrupar en los siguientes apartados:

a)Comandos relacionados con el almacenamiento digital, bien sea en disco o en cinta magnética, con instrucciones relacionadas. En este apartado vamos a tratar los comandos: BLOAD, BSAVE, CLOAD,CSAVE, LOAD, SAVE, MERGE, MOTOR, EOF.

b)Comandos que introducen datos en el ordenador por los diversos ports que éste posea, bien por teclado u otras posibles entradas, sin contar el *port* del *joystick*.
Los comandos a tratar son: INPUT, INPUT , INPUT$, LINE, INOUT, READ, INKEY$, INKEY, INKEY .

c)Comandos que envían información del ordenador a los periféricos por los distintos *ports* de salida, desde las señales que se envían al televisor hasta la más sofisticada impresora.

Los comandos a tratar son: LIST, LLIST, PRINT, LPRINT, USING, PRINT .

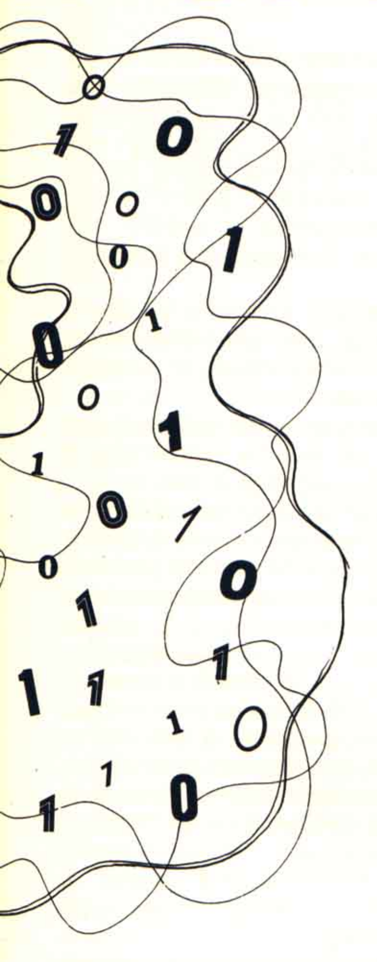
Una vez introducidos todos los comandos que vamos a tratar vamos a ir analizando poco a poco por grupos, intentando dejar claro su funcionamiento donde ejercer lo explicado.

Comandos de E/S de datos para soportes magneticos

Tanto los *diskettes* como las cintas de *cassette*, se basan en el almacenamiento magnético de datos. Aunque sea innecesario entrar en detalles sobre el tema, podemos decir que todos ellos aprovechan la susceptibilidad magnética de metales o aleaciones que hace que la llegada de datos modifiquen su estado permanentemente, de forma que al intentar volver a leer el dato es posible recuperarlo.

Para entender un poco mejor ésto,

n el MSX





El BASIC MSX es un potente lenguaje compuesto de numerosas instrucciones que convierte las operaciones más complicadas en algo muy sencillo. Esto se debe a que posee comandos específicos para cada acción, lo que simplifica y facilita bastante su programación. Sería tarea engorrosa describir las instrucciones que pertenecen a este lenguaje, pero es algo que hay que hacer si se desea profundizar en las inmensas posibilidades de este estándar.

supongamos una serie de particulas magnéticas alineadas a las que desplazamos uniformemente. Si en un momento dado introducimos un campo magnético, una perturbación, modificaremos el estado de alguna de las partículas. Esta modificación puede ser por ejemplo el cambio de la orientación de la partícula.

Si posteriormente intentamos ver la disposición de las partículas podremos ver que al llegar a esa partícula influenciada existe un cambio en la disposición respecto a las demás. Esto podría indicar cambio de información. Como sabemos, en circuitos lógicos o digitales, la información puede ser de dos tipos, positiva (1), o negativa (0), es decir, todo o nada.

Una completa gama de instrucciones de lectura y grabación de datos, permiten el acceso a la información.

Cuando se de un cambio en la información, se podrá entender que ha habido una alteración en la disposición de las partículas de información. Este es el principio del almacenamiento digital. La diferencia entre el disco y la cita no reside en ésto, sino en la forma en que se graban y recupera los datos, así, mientras que en la cinta el almacenamiento es secuencial, es decir, un bloque de datos se almacena a continuación del anterior, en un disco es totalmente aleatorio, pudiendo darse que el bloque de datos almacenados previamente, estén detrás de otros almacenados posteriormente.

Además, la diferencia podemos observarla cuando queremos recuperar los datos, ya que mientras que en una cinta hemos de leer todos los bloques de información anteriores para acceder a uno en concreto, en el disco, sólo debe leer en el directorio los sectores donde están almacenados los datos y dirigirse a él.

Las instrucciones comprendidas en este apartado ayudan al manejo de esta posibilidad de almacenamiento digital que los ordenadores poseen. No obstante, no vamos a entrar en protocolos, sino que hablaremos de las instrucciones genéricas utilizadas tanto en el almacenamiento en disco como en cinta.

Para entrar en situación, bueno es recordar que la información que queremos almacenar y recuperar podemos dividirla en dos tipos, *bytes*, y programas. Aunque en un plano más profundo es lo mismo, ya que todo son *bytes*, podemos dividirlos así de cada a nuestro uso y a las diferentes instrucciones que hay para uno u otro tipo.

Es fundamental, que no quede en el aire la diferencia entre los dos tipos, podemos decir, que *bytes* se-



FOTO 1;
La información y los datos, se pueden almacenar de las maneras más diversas, desde el cassette hasta el disco compacto, pasando por las cintas de video.

rán tablas de datos, subrutinas e código máquina, etc., mientras que un programa es lo más usual que se almacena, aunque como veremos más adelante tiene sus matices.

BLOAD y BSAVE, dos comandos para tratar los bits y bytes

Vamos a tratar en un principio, por su mayor simplicidad el almacenamiento y recuperaión de *bytes*.

Normalmente, cuando queremos salvar una parte de la memoria del ordenador, utilizamos el comando BSAVE. Pero hay que tener en cuenta que al no tener estructura BASIC, el ordenador no entenderá a partir de que posición tiene que empezar a almacenar, ni hasta que posición tiene que hacerlo, por lo que debemos especificarle en el comando BSAVE las direcciones de comienzo y final de esto.

Si estamos salvando código máquina, tenemos además la opción de poder especificar la dirección de entrada a la subrutina, a partir de la cual se ejecutará esta.

Por ejemplo, supongamos que tenemos una rutina en código máquina que ocupa 40 *bytes* a partir de la dirección 39000, cuyo nombre es 'cm' y la dirección de autoejecución es 39010 y una tabla de datos que por comodidad la almacenaremos a partir de la dirección 40000 y que ocupa 1540 *bytes*.

Si queremos almacenar esta configuración de memoria tenemos dos opciones:

Opción 1). Salvar por separado ambas partes con las siguientes instrucciones:

La rutina en C.M.:
BSAVE "CAS:CM", 39000, 39040, 39010
y la tabla de datos (suponemos que se llama 'T'):
BSAVE "CAST:T", 4000, 41540

Opción 2). Salvar ambos ficheros juntos como uno solo dada su proximidad en memoria, siempre y cuando se utilize conjuntamente la rutina y la tabla. Este método ocupa algo más de cinta en el *cassette* (evitándonos las cabeceras, etc), pero el sistema queda mucho más compacto.

Lógicamente, si la rutina no se utiliza con la misma asiduidad que la tabla, será mejor utilizar el sistema anterior que es más general. En esta opción, el comando a utilizar es:
BSAVE "CAS:CM", 39000, 41540, 39010
asumiendo que llamamos al bloque 'cm'.

En principio esta sería la disposición típica para almacenar *bytes*. Ahora podemos empezar a matizar varias cosas.

Por ejemplo, antes del nombre de rutina o de la tabla hemos puesto CAS:. Esto significa dónde se va a direccionar el almacenamiento, es decir, en este caso, vamos a almacenarlo en *cassette*, por ello ponemos CAS:. Si queremos almacenarlo en disco, pondremos A: o B; dependiendo de la unidad que utlicemos.

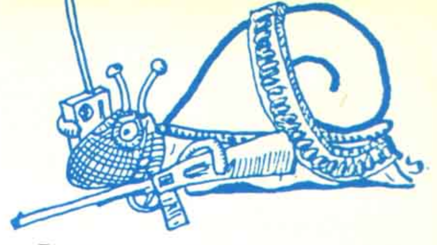
Si a la hora de escribir el comando se nos olvida algún parámetro requerido (a excepción de la dirección de autoejecución, que es opcional), tendremos un típico informe de error de MISSING OPERAND o un clásico y desesperanzador SINTAX ERROR. Se da *overflow* cuando alguno de los parámetros tenga un valor superior a 65535.

Las direcciones se pueden especificar en hexadecimal, para ello usaremos el prefijo '&H' situado delante del número.

Si durante la grabación ocurre algo extraño que nos impida seguir con la grabación, es posible interrumpirla con CTRL-STOP, dándonos, por un lado el informe DEVICE I/0 ERROR y por otro el problema de tener que volver a empezar.

Hemos dicho que las subrutinas tienen varios parámetros que se deben especificar, la dirección inicial, la direccion final y la de autoejecución. Cada una de estas se almacena en una varible con los fines que a cada ua compite. Estas son, SAVENT (FCBF) para almacenar la dirección incial, SAVEND (F87D) que almacena la dirección final. La dirección de autoejecución, si existe se guardará en la variable que almacena la dirección inicial.

En el periférico (disco o cinta) queda almacenada la direcció inicial de la subrutina (en forma de dos *bytes*,



Programas en BASIC, Código Máquina o ficheros grabados en ASCII, se tratan y manejan sin reparos.

la dirección final de la subrutina (igualmente en dos *bytes* y los dos *bytes* de autoejecución de la rutina, seguidos del contenido de la RAM que queremos almacenar.

Como los ordenadores MSX poseen Z-80, las direcciones están almacenadas en forma de dos *bytes* con el *byte* menos significativo primero. Lo que significa que si queremos almacenar la dirección &H4082 (en decimal 16514), esto se hará de forma que el valor 82 se guarde en el primero y 40 en el segundo.

Una vez tengamos el bloque salvado, habrá que tener en cuenta la posiblidad de volver a utilizarlo en futuras ocasiones. Por esta razón, nos plantearemos el problema de pasarlo de la cinta o el disco al ordenador. Para que no tengamos muchos quebraderos de cabeza podemos decir que todo lo que se almacena con la instrucción BSAVE , se puede recuperar con BLOAD. Esta es el comando que tienen los ordenadores para recuperar bloques de información, como es el código máquina o tablas de datos, estos cuando no están en forma de variables dimensionadas.

La sintáxis de esta instrucción es en todo momento similar a la de BSAVE, aunque en muchos casos no sea necesario especificar algunos parámetros.

Así para el ejemplo anterior de la rutina y la tabla, si deseáramos recuperarlos, introduciríamos la siguiente instrucción:

```
BLOAD "CAS:CM", R
```

siendo R, la inicial de la instrucción RUN, para que de esta forma se autoejecute. Esta es totalmente opcional, es decir, que sino se incluye, la rutina no se ejecutará hasta que nosotros no la llamemos externamente (principalmente con CALL). No obstate, en vez de R podemos introducir una expresión numérica (bien un número como tal o una variable que posea un valor adecuado) como dirección de autoejecución.

El funcionamiento es muy simple, lee un bloque binario inicial que lo tomará como el nombre del programa y comprobará si es el mismo que se ha indicado en el comando. Si al teclear la instrucción BLOAD, hemos omitido el nombre, es decir, hemos introducido simplemente el comando BLOAD "CAS:", se cargará el primer programa disponible en la cinta.

Si especificamos nombre de bloque (o fichero), buscará ese programa. Si el que encuentra no es el solicitado aparece el mensaje SKIP "nombre del programa", mientras que si encuentra el programa deseado, aparecerá el mensaje FOUND "nombre del programa".

El nombre cuando se trata de cargar de *cassette* debe ser de 6 caracteres, el primero de los cuales debe ser alfabético. En disco se permiten hasta 8 caracteres, aunque podemos introducir hasta 11 caracteres, pero después del octavo se colocará un punto y quedarán los tres últimos caracteres como descripción del archivo, este indicará si el programa en cuestión está grabado en formato ASCII (.ASC"), está en BASIC (".BASIC"), o es de cualquier otro formato que nosotros deseemos.

A partir del caracter 12, cualquier elemento que aparezca, no se tendrá en cuenta.

Una vez que se ha encontrado el bloque y reconocido, se introduce directamente a la zona de memoria deseada. Puede haber un error de *overflow*, cuando algún parámetro sea mayor de 65535.

La velocidad de E/S de información, manejable desde una instrucción

Otra caraterística a tener en cuenta cuanto grabamos programas, es la posibilidad de hacerlo con dos velocidades distintas.

Pues bien, los ordenadores MSX tienen dos velocidades de carga cuando trabajan con cassette, 1200 y 2440 baudios. Para entender mejor esto, vamos a definir que es un baudio, ya que sino quedará muy ambiguo. Sabemos que los ordenadores, los datos se almacenan en forma binaria y que en esta notación sólo

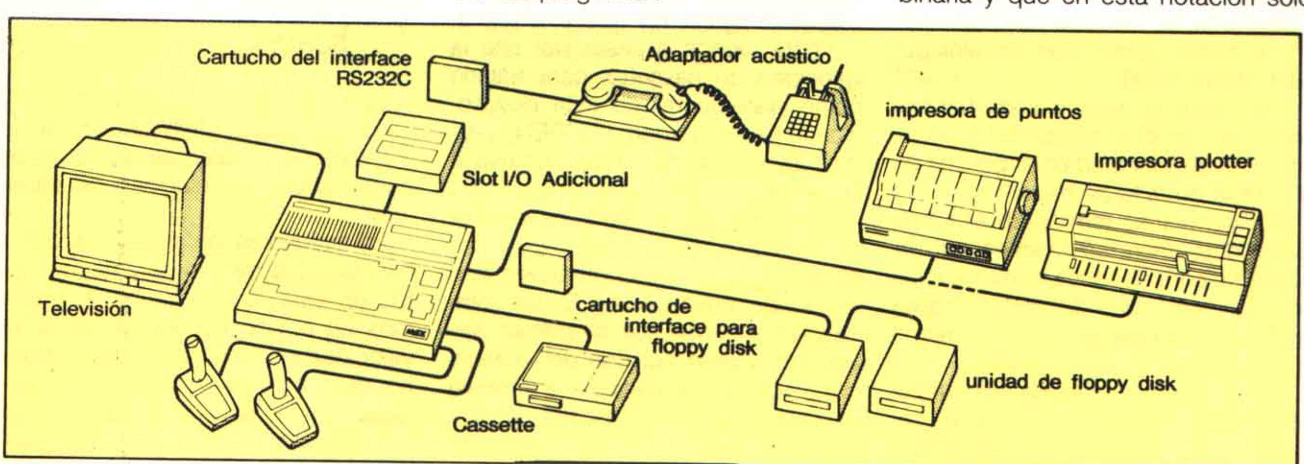


FOTO 2;
Configuración típica de un sistema MSX, donde todos los periféricos se conjuntan para formar una auténtica estación de trabajo.

hay dos signos, es decir, al transmitir los datos sólo debemos dar dos impulsos diferentes arbitrarios que correspondan a esos símbolos.

En transmisión de datos, un baudio es una de estas informaciones a transmitir, es decir, un (1) ó (0).

Luego decir que un MSX tiene una velocidad de transmisión de datos vía cassette de 1200 baudios suponer decir que transmite 1200 unos o ceros por segundo. Con esta definición puede quedar clara para cualquier persona. Para los muy puristas, diremos que los datos se almacenan en *bytes* que están constituidos por ocho *bits*, y que la velocidad en baudios corresponde al número de *bits* (no de *bytes*) que se transmiten por segundo.

Podemos decir que cuando el ordenador manda datos a un *cassette*, lo hace vía moduladores FSK con contadores variables (dos velocidades). Estos moduladores se encargan de modificar la salida del ordenador hacia un periférico que es bastante más lento que el ordenador. Son algo así, como *buffers*, zonas intermedias, que son capaces de adaptar señales.

Bién, ya sabemos que los MSX pueden transmitir datos a dos velocidades diferentes cuando trabajan con *cassette* (con discos, esta diferenciación no tiene sentido, ya que los canales de salida son distintos).

Ahora nos preguntaremos como decir al ordenador que trabaje en una u otra velocidad, lo cierto es que decirle que trabaje en la velocidad más lenta es fácil, no hay que indicarle nada, puesto que el valor por defecto es 1200.

No obstante, existen instrucciones que colocan el contador del modulador FSK a velocidad baja, porque en los MSX no falta de nada.

Existen varias formas de alterar la velocidad, ahora sólo vamos a explicar una, para después, al explicar los demás comandos de entrada/salida de datos podemos ir introduciendo otras formas.

Una manera cómoda de hacer esta modificación es utilizar el comando SCREEN. La utilización normal de este comando es algo más compleja de lo que normalmente se utiliza. Así, dentro de este comando es posible definir cinco parámetros que van a delimitar varios modos de operación para la pantalla, los *sprites*, el *cassette*, el teclado y la impresora.

La velocidad de grabación, así como el motor del cassette, se pueden controlar directamente.

El formato general de la instrucción es:

SCREEN (MODO PANTALLA), (TAMAÑO SPRITE), (SONIDO TECLADO),
(VELOCIDAD CASSETTE), (OPCION DE IMPRESORA)

Son cinco parámetros que en la mayoría de los casos no se utilizan conjuntamente, pero todos son imprescindibles. El modo de pantalla se refiere a las cuatro posibilidades que MSX permiten de representación. El tamaño de *sprite* corresponde también a las cuatro posibilidades que tiene para definirlos.

No son el objeto del artículo por lo que no encontraremos en detalle, ya que son materia de mucha discusión.

En cuanto al sonido del teclado, se refiere al *click* que se oye, si el volumen de TV lo permite, de forma que es posible inhibirlo con 0 o activarlo con algún valor distinto de cero.

Ahora vamos a pasar por alto la velocidad de cassette, para tratarlo como plato fuerte de esta descripción en el comando SCREEN, veamos ahora la forma en que se usa la impresora.

Tomando una postura egocentrista, podemos dividir a las impresoras en dos grupos: las compatibles con MSX y las que no lo son. En las compatibles con MSX, los caracteres gráficos coinciden con los del ordenador, por lo que al decirle que imprima un carácter gráfico deseado, lo hará bien. Por el contrario, si el juego de caracteres gráficos no coincide, al indicarle que imprima un carácter gráfico, no lo hará como es de esperar. Para evitar esto, con el comando SCREEN se puede indicar al ordenador si la impresora que conectamos es compatible o no con el ordenador. Si no indicamos nada, la máquina da por supuesto que lo es y listará los caracteres gráficos que deseemos, mientras que si indicamos, que no es compatibles cuando aparezca un carácter gráfico, mandará un espacio. Para seleccionar esta opción pondremos un 0 (impresora compatible MSX), mientras que si no lo es, pondremos un valor distinto.

Lógicamente, si sólo se quiere modificar uno de los parámetros dejando fijos los demás, sólo hay que poner éste pero respetando las comas, es decir, supongamos que queremos trabajar con sonido en el teclado, pondremos:



FOTO 3;
El cassette es el soporte más utilizado y la manera más barata de almacenar y recuperar la información.

SCREEN ,,1

si por el contrario queremos trabajar con una impresora de caracteres gráficos distintos a los de MSX, introduciremos:

SCREEN ,,,,1

Una vez situados en el tema y como explicación final del comando SCREEN, explicaremos de cómo se puede alterar la velocidad de transmisión.

En realidad es tan simple que como han sido los demás parámetros vistos en esta situación. Así, si ponemos en el sitio correspodiente 0, la velocidad seleccionada será de 1200 baudios, mientras que si lo que ponemos es 1, habremos selccionado la velocidad superior, de 2400 baudios. Igual que antes, si sólo queremos modificar la velocidad, dejando

todos los parámetros fijos, podemos introducir la siguiente instrucción:

SCREEN ,,,1

y habremos puesto la velocidad a 2400 baudios.

Como nota final, el valor por defecto en la transmisión de datos es 1200 baudios.

CLOAD y CSAVE, aplicable sólo al cassette

Vamos a seguir utilizando comandos en orden creciente de complejidad. Ahora le toca el turno a CLOAD y a CSAVE. Se utiliza básicamente en la grabación y recuperación de datos y programas en *cassette* solamente, mientras que los comandos BLOAD y BSAVE, permiten elegir el tipo de periférico.

En un principio, vamos a salvar programas con CSAVE. La sintáxis de la instrucción es:

CSAVE "(NOMBRE)", VELOCIDAD

Como podemos ver, aquí también tenemos otra de las muchas posibilidades anunciadas anteriormente para modificar la velocidad de transmisión de datos. Funciona a nivel de parámetros igual que en SCREEN, de forma que si ponemos 1, habremos seleccionado la velocidad de 1200 baudios, mientras que si ponemos 2, habremos selccionado 2400. Recordaremos que antes los parámetros eran 0 o para 1200 y cualquier otro valor distinto de 0 para elegir 2400.

La mayoría de los usuarios se preguntará, ¿porqué no cambiar la velocidad y grabar siempre a 2400 baudios?, puesto que al ser el doble se hará con más rapidez.

Este razonamiento es bueno, siempre y cuando el *cassette* que utilizemos este en condiciones, por el contrario, si graba mal lo cual no es nada difícil, podemos tener serios problemas para cargar y grabar programas. Por este motivo, se requiere un buen aparato o aunque sea mediocre, que esté en óptimas condiciones.

Significa esto, que la velocidad más alta habrá de evitarla y así también evitaremos problemas típicos en estos casos.

Una vez visto el problema que plantea semejante elección, especificaremos las condiciones que debe tener la sentencia para que funcione perfectamente. En un principio, el nombre que le demos al programa, debe tener como máximo 6 caracteres. Si tiene más, sólo tomará los seis primeros. Tendremos problemas con informes de error si no ponemos bien el nombre, las comillas, o un valor que no sea i el 1 ni el 2, en la opción de la velocidad.

Al empezar a grabar, hay un espacio de seis segundos en los cuales hay un silencio, y no es efectivo el CTRL-STOP.

FOTO 4;
Los diskettes se convertirán en breve plaza, junto con el cassette, en dos elementos imprescindibles para llevar un total control de los datos.

Posteriormente si lo es, pero deberemos empezar de nuevo si queremos guardar la información.

Una advertencia que a muchas personas les parecerá tonta pero que ocurre más de una vez, es que no hay que olvidarse de pulsar RECORD en el *cassette*, ya que si no, lo que habremos hecho ha sido perder el tiempo.

El comando siguiente CLOAD, tiene algunos matices más que CSAVE, ya que sirve para varias cosas. Como es fácil de imaginar, CLOAD, sirve en principio, para cargar desde *cassette* o programa que hemos salvado con CSAVE. La sintáxis general de la instrucción es:

CLOAD (?) "(nombre)"

Si empezamos por partes, vemos que es posible poner el signo (?), que va a indicar al ordenador que debe verificar o revisar el programa que se ha salvado con CSAVE. Esta instrucción consiste en comparar el programa que hay en memoria, con el que se acaba de grabar en *cassette*. Si hay algún error nos los comunicará mediante el correspondiente mensaje. A continuación tomaremos las medidas oportunas y que en la mayoría de los casos será la de volver a grabar el programa de nuevo., así hasta que funcione correctamente.

En cuanto a la selección de la velocidad, aquí no hay problema ya que el ordenador sabe a que velocidad se está grabado y automáticamente la selecciona a la hora de leer los datos.

El nombre del programa es opcional en esta instrucción si se omite cargará el primer programa que encuentre, mientras que si especificamos alguno en concreto, empezará a buscarlo, si encuentra uno que no es el requerido, aparecerá el informe de SKIP y cuando lo encuentre, aparecerá el de FOUND, cargándose o verificándo, según lo que desee.

Lo primero que encuentra el ordenador es el encabezamiento, siendo este de diez *bytes*. En él se indican las características de lo que hay a continuación, tanto el nombre como las instrucciones del programa.

Una vez que ha encontrado el programa que se busca, el ordenador hace un NEW con lo que todo lo que hubiese en memoria por debajo del RAMTOP se borra. Si queremos conjuntar dos programas, podríamos utilizar la instrucción MERGE, pero con programas salvados con CSAVE, esto no es posible. Más adelante veremos la forma de hacer esto.

La única precaución a tener en cuenta con programas grabados con el comando CSAVE, es que deben cargar con la instrucción CLOAD. El motivo lo veremos más adelante.

Antes vamos a introducir una nueva instrucción, que aunque propiamente dicha, no es un comando de entrada/salida como las anteriores, envía una señal al exterior. Nos referimos al comando MOTOR, cuya sintáxis es:

Este permite controlar el motor de un *cassette*, lo que nos da un control absoluto sobre la máquina. Esto es posible, gracias a un conector que

MOTOR (ON) (OFF)

tiene al lado de la entrada MIC. Esa conexión más pequeña, es decir, actúa como un interruptor.

De esta maneras, si queremos que el motor empiece a funcionar, pondremos MOTOR ON y cuando queramos paralo, haremos MOTOR OFF.

A primera vista puede parecer poco útil, ya que también estamos nosotros para dar a la tecla de PLAY, pero cuando desde un programa necesita salvar o cargar datos, es muy cómodo despreocuparse de controlar el *cassette*, cuando lo puede realizar el ordenador por nosotros.

Grabar o leer en cassette o diskette es operación que se efectúa, de una manera sencilla y sin complicaciones.

Además, otro posible uso, que a más de uno se le ocurrirá es el de hacer avanzar la cinta hasta una posición determinada y cargar o grabar el programa deseado.

Vamos, entonces a tratar las últimas instrucciones que podemos agrupar en este apartado, que son LOAD y SAVE.

Cuando almacenamos un programa BASIC, tenemos varias posibilidades de hacerlo:

a)En forma de *bytes*, para lo cual utilizaremos el comando BSAVE, perdiendo la estructura que da el BASIC. Este método no es aconsejable para los pocos experimentados en la disposición interna de la memoria en los MSX.

b)En forma de programa de *tokens*. Es lo más normal y la que de hecho se utiliza cuando se almacena con CSAVE. No obstante, lo primero será explicar que es un *token*.

El ordenador, en su ROM, adigna a cada comando un número y éste es el que se almacena cuando se utiliza dicho comando, es decir, por ejemplo, para un comando PRINT, la máquina le asigna un valor numérico que se grabará con la instrucción.

c)En formato ASCII, es decir, en lugar de grabarlo como números que corresponden a instrucciones, se grabarán los caracteres que forman la instrucción PRINT,incluyendo los espacios en blanco.

Esto, en principio indicaría que con este sistema ocuparía más cinta de *cassette*, pero es la única forma de poder utilizar el comando MERGE.

Se utiliza el comando MERGE se desea unir dos programas BASIC, sin perder ninguno. Lógicamente se superponen y cuando en los dos programas aparecen dos líneas iguales, permanece la que se ha cargado en segundo lugar.

La sintáxis de la instrucción es:

MERGE "(PERIFERICO: NOMBRE DEL SEGUNDO PROGRAMA)"

Es decir, para poder utilizar este comando, deberemos tener en memoria un programa BASIC al que queramos unir otro. Pulsaremos MERGE "cas:(nombre)", aunque el nombre es opcional.

Hacemos hincapié en la característica de que para ser cargado, antes tendrá que haberse cargado con el formato del código ASCII, de lo contrario esta instrucción no tendrá efecto alguno.

Una vez introducidos todos los matices, podemos dedicarnos a las funciones SAVE y LOAD. La sintáxis es como en los demas casos:

SAVE "(PERIFERICO: NOMBRE)

Para salvar un programa, aunque hayamos puesto el periférico, esto no afectará la ejecución ya que sólo es posible poner como nombre de periférico el de CAS:.

Ahora bien, si lo que deseamos es cargar un programa, inroduciremos el siguiente comando:

LOAD "PERIFERICO: (NOMBRE)", (R)

donde los parámetros introducidos entre paréntesis, son opcionales y de éstos, sólo la R es digna de mención, ya que el omitir el nombre es ya una práctica común, no sólo por parte de nuestros lectores, nosotros también lo hacemos, El poner la R al final de la instrucción, no es más que para ejecutar el programa directamente, es una forma de autoejecución.

Hay que tener cuidado de no intentar cargar *bytes* con LOAD, ya que estaremos perdiendo el tiempo, esta función debemos hacerla con la instrucción BLOAD.

Si tenemos disponible la unidad de disco, las cosas pueden variar un poco, ya que si queremos sacar o cargar un programa en un unidad, no tenemos más que escribir:

SAVE "(NOMBRE)", para salvar
y
LOAD "(NOMBRE]", para cargar, aunque estas instrucciones caen dentro del MSX-DOS, que no es el objeto de este artículo.

Existen también matices que podemos añadir cuando trabajamos con ficheros, pero no se van a tratar aquí, Entre estos, tenemos la posibilidad de cerrar los ficheros con CRTL-Z en lugar de utilizar EOF, ect.

Aquí finaliza esta extensa introducción a los comandos E/S más populares, Hemos intentado dar una idea clara de unos comandos importantes dentro de la gestión de entrada/salida de los MSX, como es el almacenamiento y recuperación de datos principalmente vía *cassette*, que es el medio más asequible de hacerlo.

Aquellos que posean la unidad de *diskette* podrán aplicar todos estos comandos con pocas variaciones a su medio de almacenamiento. De cualquier manera, con este artículo, estamos seguros de cubrir un pequeño camino en el largo recorrido que tiene por delante el BASIC del MSX.

NOTICIAS

Concurso programación Sony



La iniciativa de *Sony* debería ser un ejemplo a seguir por los diversos fabricante. El primer concurso de programación, organizado por ellos, cuenta con muchos alicientes, entre los que está el premio al ganador que será 500.000 pts.

El concurso está dividido en dos categorías; la primera es para Escuelas y la segunda para usuarios.

En el primer caso, podrán participar estudiantes y el tema a desarrollar será didáctico. El premio será de 1.000.000 pts., repartidos equitativamente por la escuela y el ganador.

En el segundo caso, participarán los usuarios del sistema, que podrán desarrollar un programa basado en cualquier tema.

Esta categoría, se caracteriza por los premios, ya que éstos varían, puesto que el ganador se embolsará, la nada despreciable cantidad de 500.000 pts. Se repartirán además, 10 premios de 100.000 pts., y habrá premios de consolación para todo el que participe.

La preocupación por el tema de la informática es manifiesta, para ello, darán clases de la materia a todos aquellos profesores que deseen asistir a ellas.



Expreso de Oriente

No podía faltar en esta sección, donde este mes las noticias son bastantes interesantes.

El mercado japonés de ordenadores MSX no hace más que crecer y crecer. Nuevos ordenadores se están presentando casi contínuamente y las aplicaciones de estos ordenadores empieza a hacerse notar.

Para empezar, el desarrollo tecnológico nipón tiene una meta: Europa.

Parece ser, que el viejo continente está en el punto de mira de los importantes fabricantes de ordenadores personales y de equipos de alta fidelidad. Como era de esperar, España, Holanda, Francia e Italia, se han convertido en el terreno donde se va a desarrollar una importante lucha en pos del mercado.

Sin embargo, la guerra sin cuartel, tendrá lugar en el mercado americano. Indudablemente, puede que con resultados desiguales, pero las modificaciones a las que están sometiendo los diversos ordenadores que se esperan enviar para su pronta comercialización, van a dar que pensar.

Una unidad de *diskette* y *software* incorporado, 128K memoria RAM de video, así como 128K de memoria para el usuario. También se le añaden interesantes opciones, van a dar mucho trabajo a los ingenieros de **Commodore**, si desean combatir el gigante amarillo.

Z800, ¿sucesor del Z80?

La gran incógnita en el desarrollo de la segunda versión del MSX se va a entrar en la Unidad Central de Proceso. Z800 y Z8000, son las dos posibilidades con las que se cuenta a la hora de hacer un ordenador de 16 *bits*.

Esta versión, totalmente compatible con el Z80, permitirá al ussuario y a los fabricantes, optimizar el rendimiento de estas computadoras. A unos, por su fácil manejo, a otros, por su flexibilidad y potencia.

Además de contener instrucciones nuevas, trabaja a una velocidad bastante elevada y tiene el mismo juego de instrucciones que su predecesor, el Z80.

Por el momento, este *chip* no se va a comercializar. Las razones siguen siendo un misterio, ya que este *chip* se anunció hace varios años y aún no hemos visto ejemplar alguno. De cualquier manera, *Zilog*, fabricante de esta versión más potente del Z80, cree que a finales del presente año, habrá información técnica y manuales sobre este *chip* y que las primeras unidades aparecerán a principios o mediados del año 1986.

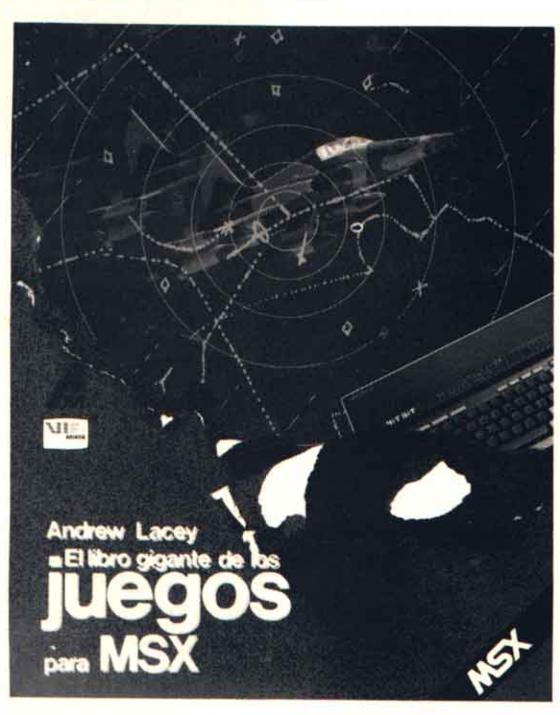
Anaya Multimedia y sus libros para MSX

Actualmente, la bibliografía existente para los usuarios del sistema MSX, deja mucho que desear. Hay pocos libros que puedan ayudar al usuario medio a defenderse de las pegas que impone un lenguaje completo y nuevo como es el BASIC MSX.

Anaya Multimedia entra en el campo del MSX con un libro, que no basa su contenido en programas de aplicación, pero que puede convertirse en una ayuda inestimable a la hora de ver casos reales.

La primera entrega, el **el Libro gigante de los Juegos para MSX,** no es una guía práctica de los comandos del MSX, pero aunque sólo sean juegos, éstos vienen tan bien comentados que en momentos pueden ser más útil que cualquier otro manual.

Los programas se comentan por secciones y se explican tanto las variables utilizadas como su significado.



Serma y Konami: la creación de software

Serma, conocida empresa en el ámbito del *software*, es el distribuidor en exclusiva para España de los juegos de otra importante empresa; **Konami**. Expresamente dedicada a la creación de *software* para ordenadores MSX, **Konami** se introduce en nuestro país para ofrecer esos juegos que tan famosos nombre tiene más allá de nuestras fronteras.

Los programas vienen en el formato de cartucho ROM, característica que agradecerán los usuarios al facilitar y simplificar el proceso de cargar un juego en el ordenador. El lanzamiento de estos juegos es inminente. Los primeros en salir serán el conocido Hyper sport, Yir ar Kung-Ku, Tennis y Circus Charlie. Los títulos irán apareciendo paulatinamente y los usuarios tendrán la posibilidad de comprobar la gran calidad de estos juegos en la feria del **Sonimag** y en la Feria-micro en el **Corte Inglés.**

SOFTWARE

Programa: Generador de Sprites.
Tipo: Juego
Distribuidor: ACESA
Formato: Cassette

¿Qué es un *sprite*?. La palabra quizás nos parezca un poco desconocido pero que sin duda hemos visto muchos de ellos. Una definición acertada y comprensible sería: una serie de puntos referidos a unas coordenadas situadas en el espacio cartesiano, en conjunto formarían un dibujo o un mensaje que asociaríamos en nuestro interior a un significado.

Muchas veces hemos visto dibujos generados por un ordenador y nos hemos vuelto locos pensando cómo se puede hacer eso (desde luego es para pensarlo), pero hoy en día, todos los amantes del dibujo, la perspectiva, etc., a pequeña escala podremos tener un gran elemento de ayuda con este programa "Generador de *Sprites*".

Siempre nos hemos visto frente a juegos que por decir de alguna manera, "nos lo daban todo hecho", y nosotros con ayuda de nuestros cursores o *joytick*, jugabamos limitando en cierto modo nuestra capacidad creativa.

Aquí podemos ser jugadores y creadores a la vez. Creadores porque se nos permite en la modestia del programa explayar nuestros poderes o facultades creativas y aún más de diseño o dibujo, y jugadores porque ¿quién no se divierte creando y desarrollando sus ideas y fantasias?.

Generar un *sprite* con este programa es muy fácil de realizar, observemos la pantalla un cuadrado de 16 × 16 casillas, que rellenaremos en orden y medida a nuestro gusto, es decir, teniendo en cuenta el *sprite* que vamos a generar. Este se irá formando porque la casilla o casillas que vayamos designando se cambiarán de color.

Aparecerá en la casilla (1, 16), un cuadrado de color, que iremos moviendo a lo largo y ancho del cuadrado, es decir, vertical y horizontalmente. Para ir rellenando casillas bastará que una vez colocada la casilla móvil en el lugar adecuado pulsemos la



barra espaciadora, y ese cuadrado formará parte de nuestro *sprite*. Si por el contrario nos hemos equivocado, y ese no era el sitio adeecuado, bastará con pulsar la tecla designada en nuestro ordenador por las siglas "BS", junto con la barra espaciadora en el cuadrado que deseemos borrar.

Terminaremos nuestro *sprite* y a la vez nuestra obra al pulsar "return". Durante todo el proceso de creación de nuestro *sprite* tendremos un cuadrado de menores dimensiones que nos irá reflejando la figura que vayamos creando, para visulizarlo de una forma más clara, la síntesis del *sprite*.

Una vez terminado nuestro *sprite*, los puntos de las coordenadas utilizadas servirán en un futuro como instrucciones si deseamos volver a generarlo.

Debajo de estas instrucciones, se imprime la pregunta, ¿Quieres pasarlo a impresora? (S/N), teniendo con esto la posibilidad de crear un amplio archivo de *sprite*, ya que al tener los comandos para su realización, podremos recrearlo siempre que queramos.

Con este programa podremos generar tantos *sprites* como necesitemos, y con una gran capacidad de singularidad reflejo de lo que nosotros seamos capaces de generar.

Más de un juego de diversión es un juego didáctico, pero no cabe duda que el entretenimiento está asegurado.

Puntuación:
Presentación: 6
Claridad: 8
Rapidez: 7

Programa: Panic Junction.
Tipo: Juego
Distribuidor: E.M.S.A.
Formato: Cassette

¿Quién más de una vez no ha subido a un tren, y ha disfrutado todo el trayecto viendo pasar grandes mostruos de acero a velocidades insospechadas?, o ¿quién más de una vez no ha jugado con sus máquinas, vagones, a ser jefe de estación haciéndolas pasar por los caminos más raros y circunstancias más extrañas?

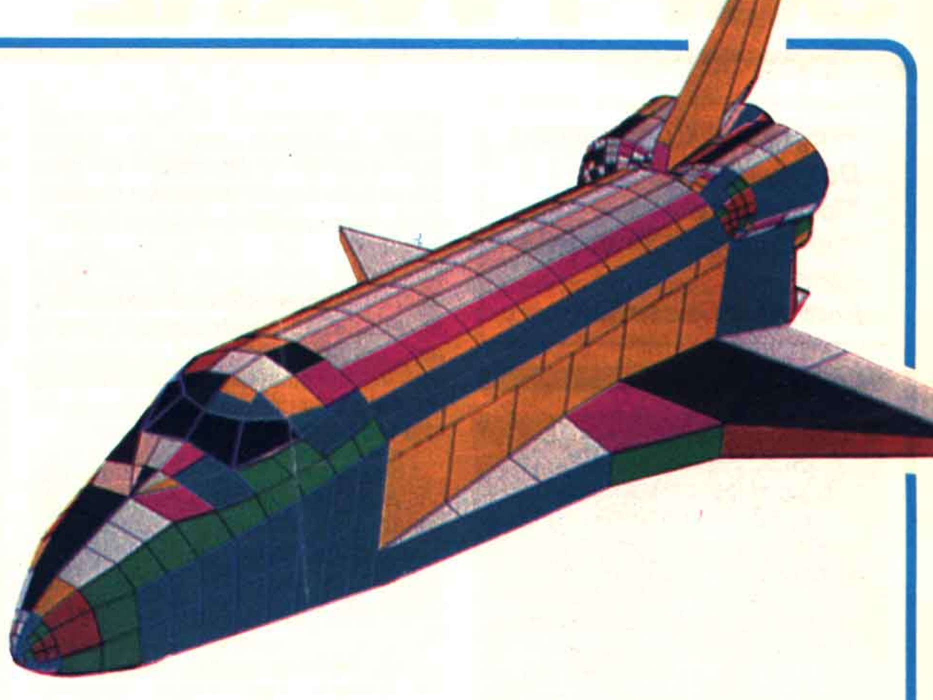
Cuando miramos las vías férreas, nos parecen un entramado sin fin que se confunde en nuestras mentes, pues algo más fácil que todo esto aunque de mayor rapidez y dinamismo nos ofrece este programa.

No es un juego de complicado manejo, aunque nuestros objetivos principalmente son tres. En nuestra pantalla, aparece un primer cuadro superior por el que van circulando locomotoras y aquí surge un pequeño problema que es que no siempre van circulando por la vía que deberían.

Un segundo cuadro de menores dimensiones que nos señala el número de locomotoras con las que en ese momento vamos a jugar y la dificultad que presentan, es decir un

croquis o la visión general para que podamos coordinar nuestros movimientos y saber a que vía debemos hacer que se dirija la locomotora que circula en una vía ya ocupada.

Y un tercer cuadro en el que más atención debemos de mostrar. Aquí nos encontramos nosotros como "salvadores" de que no ocurra una catástrofe. Nos encontramos metidos en un cuadro que a primera vista parece un laberinto, pero no entraña mayor dificultad que la rapidez y agilidad mental. Dentro del cuadro hay un número definido de palancas que corresponderán a los trayectos de vía que permiten el traslado de una locomotora a otra vía para así subsanar un choque. Nos moveremos a lo largo y ancho de este habitáculo, ayudados por nuestros cursores, o también con nuestro joystick. Una vez que lleguemos a la palanca que deseamos mover tendremos que pulsar la barra espaciadora y con nuestros cursores superior e inferior dar la orden para que así se desplace la locomotora a una u otra vía, quiere esto decir que si nosotros giramos la palanca hacia arriba, la locomotora cambiará su trayecto al de la vía superior; si por el contrario giramos la palanca hacia abajo, descenderá de vía. Si necesitamos ir a otra palanca para que nuestro guarda–agujas pueda evitar otro choque, bastará volver a pulsar la barra espaciadora y quedará libre para moverse en cualquier dirección que deseemos.



Tal vez resulte a primera vista un juego monótono, pero cuando entre en él, y vea sus propias características, es decir, las sensaciones que a usted le ofrezca, verá que es un juego en el que tendrá que adelantarse al ordenador. El juego terminará cuando hayamos permitido que choquen cinco locomotoras, pero confiamos en sus reflejos para que esto no ocurra.

En su formato es un juego agradable de ver, ya que tiene muchos puntos para concentrar vuestra visión; las locomotoras, el croquis de funcionamiento de éstas y nosotros.

El color está muy conseguido y en conjunto es un juego muy singular.



Puntuación:
Presentación: 8
Claridad: 7
Rapidez: 8

SOFTWARE

Programa: Contabilidad Doméstica
Tipo: Juego
Distribuidor: Dimensión New
Formato: Cassette

Dentro de las diversas formas en que podemos llevar un orden clasificatorio de nuestros quehaceres domésticos, nos encontramos con este programa, pensando para que podamos aprovechar y utilizar nuestro ordenador, con un fin tan importante como es llevar la contabilidad, es decir nuestra contabilidad casera.

Hoy en día, un elemento tan importante como es atender de manera específica nuestra economía se vé informatizado a un nivel que propiamente puede hacer y dar las satisfacciones de los amantes del equilibrio y la información.

La utilidad de la contabilidad de todos es conocida, pero vamos a hacer referencia a las características de este programa y a los medios y generalidades a los que está referido.

LLevar bien una economía doméstica hace necesaria la presencia de un programa el cual nos pueda ayudar y a la vez distraer, sirviéndonos en los fines a los que haya sido destinado.

Ese sector dedicado a la pequeña empresa también puede encontrar aquí, un fiel sustituto de las personas dedicadas a esta ciencia, y además conseguir a la vez una información detallada y concisa la cual no se escape a nuestro entendimiento.

El tiempo real en que este programa se mueve, es el tiempo que dura un ejercicio económico, es decir, si usted, ha decidido comenzar su cotabilidad doméstica, hoy día 1-junio-85, este programa le registrará todas las entradas y salidas efectuadas en el período de un año a contar desde la fecha antes citada, las cuales serán recogidas en una cinta *cassette* que usted dispondrá para ello.

El aspecto interno que destacamos, es una terminología dirigida al carácter doméstico de nuestro ordenador. Debemos tener en cuenta que nuestras posibilidades de relacionar exteriormente nuestra contabilidad, se ve favorecida por los recursos con que contamos en el programa, es decir, la opción de poder remitir nuestros movimientos hasta un total de nueve cuentas bancarias, permitiéndonos, en el momento que los solicitemos, en el momento rio y el saldo correspondiente a la fecha indicada.

Se le permite a su vez indicar hasta veinte conceptos diferentes de entradas y salidas y el interruptor hasta 250 registros de movimientos, además de las nueve cuentas bancarias ya citadas, todo ello por fichero.

Obviamente podría obtener tantos ficheros como desee, teniendo el propio programa un opción especializada para la lectura y grabación de los mismos.

```
DimensionNEW CONTABILIDAD PERS. MSX
     1.-ENTRADA DE DATOS
     2.-BUSQUEDA DE DATOS
     3.-ANULACION DATOS
     4.-BALANCE
     5.-PREVISIONES
     6.-DATOS BANCARIOS
     7.-SALDOS BANCARIOS
     8.-GRABACION FICHERO
     9.-LECTURA FICHEROS
     0.-DEFINICION CONCEPTOS
   indique que opcion desea
```

Todos estos pasos se irán consecutivamente haciendo de manera que no le resulte complicada su ejecución el programa le irá indicando en todo momento los pasos a seguir.

El menú principal nos presenta 10 opciones de 1 al 0, para que nosotros introduzcamos en cada uno de ella los registros que queramos guardar o anotar.

En la opción 0 se nos capacita para nombrar hasrta 20 conceptos forma que tendremos de denominar aquellos campos específicos que queramos llevar más regularmente en nuestra economía, para su ejecución debemos tener apretado el *"caps lock"*, el cual nos permitirá reflejar en la pantalla las instrucciones de mayúsculas y a su vez la posibilidad de reflejar en el ordenador las entradas y salidas de los recibos o gastos remitido a cada uno de los conceptos.

El menú anteriormente citado se nos presenta con diez opciones las cuales vamos a enumerar señalando sus caracteristicas más importantes.

Le permite dar nombre hasta a 20 conceptos, aquellos que usted disponga para llevar más regularmente su contabilidad.

A su vez en la parte inferior de la pantalla, se indican tres opciones, una primera opción para definir conceptos, una segunda que los modifica y una tercera para volver al menú principal. Una vez tenga definidos los nombres de los conceptos, le irán permitiendo registrar todos los movimientos a nivel de entrada y salida. Entonces utilizaremos la opción nº 1, denominada entrada de datos, en la cual aparece escrito el orden en que tiene que introducir los datos y el lugar adecuado para ellos. Estos datos hacen referencia, al día, mes y año (DDMMAA), nº de c/c, cantidad o volúmen, etc.

La opción nº 2, denominada, "búsqueda de datos", nos permite acceder a cualquier dato, que hayamos registrado anteriormente y los cuales podremos obtener por fecha, concepto, cantidad y memo (referencia del concepto), teniendo también la posibilidad de obtener un listado de todos ellos para hacernos una idea de la situación.

Anulación de datos, opción nº 3, como su propio nombre indica, esta nos capacita para poder anular o modificar algún error en la entrada de datos y así borrarlo del fichero principal.

Opción nº 4, Balance: Es una de los apartados más importantes porque nos permite establecer una visión de los datos registrados y el poder obtenerlos por impresora.

Opción nº 5, Previsiones: Nos permite dotar provisiones referidos à los datos y balances correspondientes.

Opción nº 6, Datos bancarios: Nos hace referencia a las nueve cuentas bancarias que podemos utili-

zar para llevar nuestra contabilidad y las cuales podremos conseguir saber su saldo en cualquier momento.

Opcion nº 7, Saldo bancarios: Referida integramente a la anterior, nos ayudará a obtener el extracto de la cuenta que elija pudiendo además regularizar el saldo si lo desea.

Opcion n.º 8, Grabación de ficheros: Nos permitirá resguardar todos los datos registrados en una *cassette* que usted habrá dispuesto para ello.

Opción nº 9, Lectura de ficheros: Podrá con esta opción leer de la cinta un fichero, aunque perderá los datos que posea en la memoria del ordenador.

Estas son las caracteristicas estructurales del programa, las cuales aplicadas a una contabilidad doméstica pueden ser de gran utilidad para conseguir llevar una economía equilibrada y al día.

Puntuación:
Presentación: 8
Claridad: 6
Rapidez: 8

Programa: O'Thelo.
Tipo: Juego
Distribuidor: Dimensión New
Formato: Cassette

Todos aquellos que hayamos pasado largos y entretenidos ratos jugando, sabemos las principales características de este juego, es decir su infraestructura, las reglas en las que se basa, los momentos de duda, concentración, nerviosismo y como no, la capacidad de desarrollo, táctica y previsión frente a los duros movimientos y situaciones angustiosas en las que más de una vez nos han puesto nuestros compañeros o amigo de partida.

El hombre siempre ha sido precursor de avances, nuevos prototipos y ha desarrollado su capacidad resolutiva tanto en la creación de instrumentos de mejora para la sociedad, como elementos para satisfacer y cultivar su interior en sus momentos de ocio, aquí aparece con un papel preponderante y con el futuro a su favor, el *software*.

Dentro de los numerosos programas que inundan el mercado aunque cada día los nuevos programas son una superación de los anteriores, encontramos aquellos juegos que una vez fueron moda y que hoy forman un conjunto de clásicos, indispensable para cualquier videoteca de *software*, entre los que se encuentra O'Thelo.

Las personas en nuestro "ego", experimentamos gran satisfacción cuando vencemos a nuestro adversario, habiéndonos valido de nuestra astucia y habilidad, en este juego podremos experimentar todo ese tipo de sensaciones e incluso nuevas al sumirnos en el mundo de los ordenadores y observar lo necesaria y satis-

factoria que resulta su compañia.

El carácter general que caracteríza a todos los juegos de reflexión y dinámica estructura se vé individualizado en este juego tan singular, reflejando como está claro las posibilidades que tenemos de elegir adversario: persona u ordenador.

Es juego creado con una amplitud de miras para distraernos, comenzaremos cargando el juego en nuestro ordenador y éste realizará un pequeño estudio de la situación, es decir:

1° Si nosotros deseamos jugar con el ordenador (implacable), o
1° si jugaremos con otra persona como adversario.

Si optamos por la segunda opción mediremos nuestra capacidad de estrategia frente a nuestro compañero y seguro que pasaremos un rato o una tarde de los más agradable.

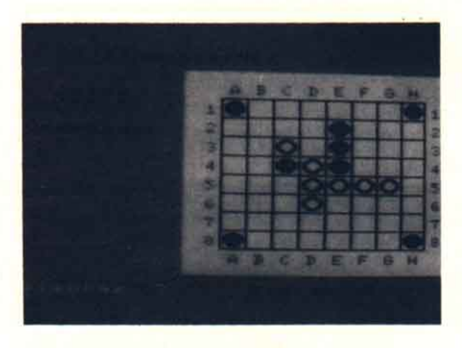
No obstante si aceptamos a nuestro ordenador como compañero de diversión y distracción, sin duda no hemos elegido mal, pero ... pero será un adversario al cual se le pueda hacer cambiar de opinión con una agradable locución o una inminenete sentencia, pues nuestro ordenador jugará con el tesón, implacabilidad, sangre fría y astucia de la que pudiera disponer el mejor estratega del mundo.

Realizará un segundo estudio sobre nuestras características en lo concerniente al juego, bien, si hemos jugado muchas veces, es la primera vez o si somos unos expertos dignos de mención en el O'Thelo.

Se presentarán nueve opciones que medirán nuestros conocimientos, estos irán desde el nivel:

1° Muy excepcional, hasta ...
9° Excepcional.

Deberemos saber cuales son



nuestras posibilidades pues el ordenador no nos dará alternativa para volver atrás ya que jugará siempre en cualquier nivel de los que estemos, al máximo de sus posibilidades. Tampoco se nos permitirá hacer trampas, pues el mismo ordenador será arbitro y señalará "jugada ilegal" cuando hayamos cometido alguna falta.

La estructura de la pantalla es muy fácil, en el margen lateral derecho, nos aparece un cuadrado de 8×8 casillas, en donde irán colocando las fichas a lo largo de la partida, en el lado lateral opuesto nos dirá el turno del jugador al que le toca mover y también nos dirá cuando realicemos una "jugada ilegal".

Abajo en el margen inferior señalará las fichas con las que contamos al inicio de la partida y así a lo largo de ella. Esto se verá reflejado en la pantalla por "blancas" (que serán las nuestras siempre que juguemos contra el ordenador) y "negras", con las que jugará bien el ordenador o bien nuestro compañero, pero al fin y al cabo nuestro adversario.

Si nuestro nivel de juego, es el normal, tendremos el mismo número de fichas que nuestro adversario, si nos encontramos por debajo del nivel normal, nos dará unas fichas de ayuda y si ocurre el caso contrario, que estemos por encima del nivel normal seremos nosotros quienes demos fichas de ayuda al ordenador.

El juego comienza presentándonos dos fichas negras y dos blancas (en el caso del nivel normal), situadas en diagonal, en el centro del cuadrado. Se trata de conseguir rellenar el cuadrado con el mayor número de fichas nuestras posible, pero para esto debemos de conocer las reglas del O'Thelo. Conseguiremos "comer" las fichas de nuestro adversario siempre que coloquemos dos de nuestras fichas a los laterales de las suyas (que pueden ser una o varias). Las posiciones de nuestras fichas, bien vertical, horizontal o diagonal a las fichas de nuestro adversario.

Se nos avisará de que estas fichas han sido comidas porque cambiarán al color de nuestras fichas y también porque se verá reflejado en los marcadores.

Colocaremos las fichas en la casilla que queremos pulsando primero la letra de la colúmna y después el número de la fila.

La característica del juego le harán pasar a un estado de gran concentración y a la vez de diversión.

Es un juego creado especialmente para los "amantes del O'Thelo" pero a su vez deja cabida a todas las personas que quieren iniciarse en este ingenioso y entretenido juego.



Puntuación:
Presentación: 8
Claridad: 9
Rapidez: 8

LIBROS EN CASTELLANO PARA TU ORDENADOR

AMSTRAD SPECTRAVIDEO sinclair ZX Spectrum y QL MSX



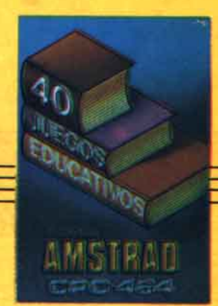
*Manual de Referencia Basic del Program. AMSTRAD.
La más autorizada y completa guía para programar en Locomotive Basic.
3.400.— Pts.



*Sensacionales Juegos AMSTRAD.
Listados completos de 27 estupendos juegos de muy diversos estilos.
1.950.— Pts.



*Programando con AMSTRAD.
Fundamental para el usuario principiante. Ameno y repleto de ejemplos.
2.400.— Pts.



*40 Juegos Educativos AMSTRAD.
Listados completos (matemáticas, geografía, música, etcétera) para aprender divirtiéndose.
1.950.— Pts.



*Lenguaje Máquina... AMSTRAD.
Ideal para iniciarse en el código máquina del 780 Y EN EL SISTEMA OPERATIVO DEL AMSTRAD.
2.100.— Pts.



*Interferencia Artificial AMSTRAD.
Convierta su AMSTRAD en un compañero inteligente.
1.500.— Pts.



*Sonidos y Música AMSTRAD.
Programe música y efectos sonoros y convierta su AMSTRAD en un sintetizador.
1.200.— Pts.



*Programación Básica spectravideo.
Imprescindible para iniciarse en el dominio de las estructuras fundamentales del Basic.
1.800.— Pts.



*Programación Avanzada SPECTRAVIDEO
Para "saber más": ficheros, subrutinas, gestión de errores, funciones definibles, etcétera.
2.400.— Pts.



*Código Máquina SPECTRUM
Las instrucciones fundamentales del Z80 para iniciarse en el código máquina.
2.100.— Pts.



*Los 20 mejores programas.
Selección de excelentes programas en Basic.
1.800.— Pts.



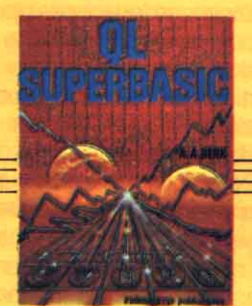
*Programación avanzada.
Subrutinas, trucos y análisis para mejorar tus programas.
2.200.— Pts.



*Las 40 mejores SUBRUTINAS
Las más útiles rutinas en código máquina reunidas en un sólo volumen.
1.950.— Pts.



*Programando con QL.
Texto introductorio, claro, útil y ameno.
1.950.— Pts.



*QL Superbasic.
Un curso avanzado para dominar el excelente Basic de tu QL.
1.950.— Pts.



*Programando con MSX Basic.
Curso completo y detallado, con numerosos ejemplos prácticos.
2.200 Pts.



*El libro de juegos MSX.
Listados completos y comprobados de 21 excelentes juegos.
1.900.— Pts.

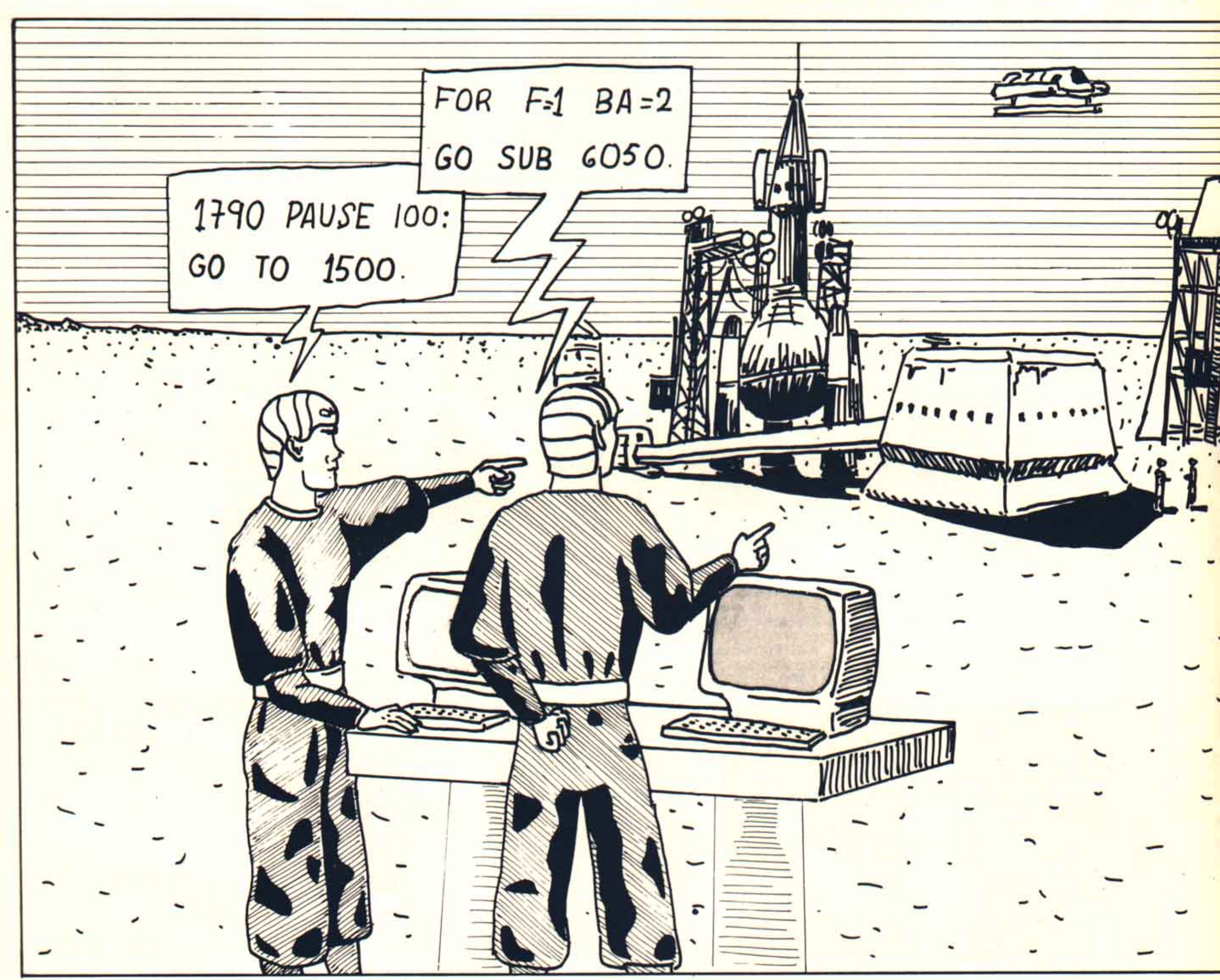


PUBLICACIONES

Avda. del Mediterráneo, 9
Tels. 433 45 48 - 433 48 76
28007 MADRID
Delegación en Cataluña:
Tarragona, 110 - Tel. 325 10 58
08015 BARCELONA

DE VENTA EN El Corte Inglés
Y TIENDAS ESPECIALIZADAS

Comparativa entre los ordenadores



Cada ordenador personal tiene sus defensores y sus detractores. Características que a unos gustan, pueden no ser del agrado de otros. Es lógico y normal que cada usuario defienda a su ordenador personal con uñas y dientes, basándose en criterios a veces nada convincentes.
Expondremos las características del BASIC MSX, a la vez que lo comparamos con el lenguaje de los ordenadores más populares del mercado, que como ya supondreís son el ZX Spectrum y el Commodore 64.

¿Quién no ha oído hablar del ZX Spectrum o del Commodore 64? En el creciente mercado español, se han introducido múltiples ordenadores personales, algunos con mejor suerte que otros. Todos ellos tienen sus seguidores particulares, aunque muchos de los cuales lo compraron por ser la novedad del momento.

Pero el tiempo paso y el ordenador personal se está convirtiendo no sólo en el compañero ideal para los ratos de ocio, sino también en ayuda imprescindible en el hogar. Ya no se matan marcianos intergalácticos, ni comecocos, ni bichos parecidos, también podemos escribir artículos o

BASICS de los más populares



cartas con un tratamiento de textos, llevar una agenda completa o incluso la contabilidad de la casa. Todos estos aspectos son los que caracterizan a esos ordenadores, personales, es decir, la posibilidad de tener una máquina completa y versátil, y fácil de programar.

En lo que respecta a la versatilidad de los ordenadores, nadie dudaría si dijésemos que son capaces de realizar las taréas más inverosímiles, por muy difícil que ésta sea. Ahora bien, en el tema de la programación la cosa cambia.

Comprar el MSX, ZX Spectrum (en el artículo, haremos referencia siempre al Spectrum de 48 K, aunque sea igual que el de 16 K) y Commodore 64, y dar una opinión a favor de tal o cual ordenador no es nuestro menester. Destacaremos las particularidades de las instrucciones de cada ordenador y dejaremos que sea usted, lector, quién dé la última opinión.

Además, esperamos que este artículo le sirva para discurrir, con sólidos argumentos, la necesidad de que sus amigos se compren un MSX.

El BASIC es un lenguaje de programación que está al alcance de todos. Como es lógico, la dedicación es la mejor manera de obtener la experiencia y los resultados necesarios que nos permita utilizar el ordenar hasta el límite de sus posibilidades.

La facilidad con que unos aprenden antes que otros, a veces no es cuestión de dedicación, sino de los complicado que puede ser o no la programación en el ordenador.

Aquí topamos con el primer inconveniente. Así como en el Zx Spectrum y en el MSX, los comandos e instrucciones son fáciles y sencillas de realizar, en el Commodore podemos comprobar lo complicado de tal operación. Pero no entraremos en detalles y explicaremos más las diferencias existentes entre los distintos tipos de BASIC que presentan.

Los comandos básicos, que se



FOTO 1:
Los tres ordenadores más populares enfrentados en un exámen

pueden hallar en los tres ordenadores, son los siguientes:

- — IMPUT
- — GO TO
- — PRINT
- — READ
- — DATA
- — RESTORE
- — GO SUB
- — FOR...TO...NEXT...STEP
- — RETRUM

Existen más comandos, pero los que están implementados en uno no lo está en el otro, por lo que veremos estos casos aislados a continuación. Además, cada fabricantes ha dotado a su ordenador con unas características y posibilidades que el resto no tiene.

Pero nuestra lista de comandos, por el momento, está vacía. Sólo contiene aquellas instrucciones que podemos encontrar, en estos y otros ordenadores más.

Veamos pues, cuales son las instrucciones que tienen una función similar en unas máquinas y otras. Dividiremos este estudio en apartados, siendo la primera sección la dedicada a las instrucciones de Entrada/Salida de datos.

Pero antes de entrar a comentar la materia, haremos una aclaración. El ZX Spectrum tiene el *interface* que le permite crear ficheros secuenciales. Este caso no se ha considerado, por dos motivos; el primero de ellos es que la mayoría de los usuarios no poseen dichos *interface* y segundo, esas instruciones como OPEN , CLOSE , etc, funcionana si y sólo si, se ha conectado el *interface* I al ordenador. Si se aplica directamente veremos el mensaje de error correspodiente.

¿QUIEN HA DEJADO AQUÍ ESTOS BASICS?

De los lenguajes de programación BASIC, vistos hasta ahora, el del MSX, viene a ser la opción más completa y sencilla de las que hay.

Instrucciones de Entrada/ Salida

Para el manejo del *cassette*, nos encontramos con un buen repertorio en cualquiera de los ordenadores. Aunque el MSX se lleve la palma ya que incorpora un completo juego de instrucciones tales como:
CSAVE, CVLOAD, CLOAD?, BSAVE, BLOAD, OPEN, "CAS:" PRINT e IMPUT , etc, para gestionar la entrada y salida de datos de dicho periférico.

Las instrucciones CLOAD y CSAVE, equivalen a las instrucciones LOAD y SAVE del Spectrum y Commodore, mientras que CLOAD?, equivale a la instrucción VERIFY. El Spectrum no dispone de las instrucciones OPEN PRINT , e IMPUT , para el manejo del *cassette*. Estas en cambio, si están en el Commodore 64.

Siguiendo con los soportes magnéticos, entraríamos a comentar las instrucciones que controlan el *diskette*. En este caso, nos encontramos con que el Commodore y el MSX, pueden manejar la unidad de *diskette* directamente. En el primer caso, al final de la instrucción SAVE o LOAD, se indica, mediante un número, si la operación la vamos a realizar en *cassette* o en *diskette*, mientras que en el segundo caso, basta con indicar la instrucción SAVE o LOAD. El Spectrum no posee instrucciones para controlar *diskettes*, aunque el mercado ofrezca unidades como opción adicional al *cassette*, ya que aquel anula totalmente la posibilidad de contar con el microdrive.

En orden de importancia, veamos a continuación las instrucciones que manejan la impresora. Aquí, tanto el Spectrum como el MSX, tienen incorporado las instrucciones LPRINT y



FOTO 2:
MSX, con su BASIC de Microsoft, es de los que mejor futuro puede esperar.

FOTO 3:
El Commodore, posee uno de los BASICs más complicados y difíciles de entender.



LLIST, para listar parte o totalidad de un programa. Además el Spectrum tiene el comando COPY, que permite obtener una copia de pantalla por impresora. Este comando no lo tienen el MSX ni el Commodore, lo que significa que para copiar de pantalla, es necesario realizar una rutina en código máquina y aplicarla.

Para leer información del teclado, cada ordenador tiene una instrucción. En el MSX y Spectrum, esta es INKEY , mientras que en el Commodore 64, la instrucción es GET. En todos, realiza la misma función, lee el teclado para comprobar si se ha pulsado una tecla.

En lo que se refiere a instrucciones que controlen directamente los *joysticks*, podemos decir, que el MSX aventaja a todos sus competidores. Ninguno de los ordenadores en cuestión tiene instrucciones BASIC, que permitan tener un control total y absoluto del *joysticks*. Entre las diversas instrucciones que se ofrece, podemos destacar instrucciones del tipo:

STICK, ON STRIG GOSUB, SRIG, etc. Con ellas se pueden conocer el estado de los *joysticks* y si se ha pulsado el botón de disparo. En el Commodore, esta operación se realiza mediante POKEs y el Spectrum carece de toda instrucción (además del port de *joysticks* que permite un control de estos periféricos).

Por último, en este apartado, veremos las instrucciones que hay en cuanto a programación se refiere.

Entre el grupo de comandos de entrada/salida que hay, los tres ordenadores incorporan las instrucciones INPUT y PRINT, lo que permiten una comunicación directa ordenador-usuario. Pero además de ellas, el MSX y spectrum, añaden a la opción INPUT otra variante, LINE, que completa el juego de posibilidades. Aunque si quiere más, el MSX implementa la opción USING en el comando PRINT que le permitirá formatear los datos de salida.

Lógicamente y debido al tiempo que llevan el Commodore y el Spectrum en el mercado, han ido apareciendo periféricos adicionales que

```
Listado 1

1000 REM PRUEBA 1
1010 PRINT "P"
1020 FOR K=0 TO 1000
1030 NEXT K
1040 PRINT "F"
1050 END
```

```
Listado 2

1000 REM Prueba 2
1010 PRINT"P"
1020 K=0
1030 K=K+1
1040 IF K<1000 THEN 1030
1050 PRINT "F"
1060 END
```

```
Listado 3

1000 REM Prueba 3
1010 PRINT"P"
1020 K=0
1030 K=K+1
1040 A=(K/K)*K+K-K
1050 IF K<1000 THEN 1030
1060 PRINT "f"
1070 END
```

```
Listado 4

1000 REM Prueba 4
1010 PRINT"P"
1020 K=0
1030 K=K+1
1040 A=(K/2)*3+5-5
1050 IF K<1000 THEN 1030
1060 PRINT "f"
1070 END
```

```
Listado 5

1000 REM Prueba 5
1010 PRINT"P"
1020 K=0
1030 K=K+1
1040 A=(K/2)*3+4-5
1050 GOSUB 1090
1060 IF K<1000 THEN 1030
1070 PRINT "f"
1080 END
1090 RETURN
```

van complementando el ordenador. Así podemos ver como el *Interface* I es la mejor opción a la hora de tener información accesible y en un tiempo nada despreciable, mientras que el *Interface* II, pemite conectar cartuchos ROM dos *joysticks* al ordenador. Pero esto, al no ser parte del ordenador no entra en este estudio.

Pasemos a continuación a las instrucciones de control gráfico.

Instrucciones de control gráfico

Sin lugar a dudas, hemos llegado a la parte más curiosa e interesante de las que hay.

La gestión de gráficos es posible gracias a que tanto el MSX como el Commodore, utilizan un controlador de video, que se encarga de gestionar la pantalla.

En el Commodore, no existen instrucciones gráficas propiamente dichas, ya que todo lo relacionado con gráficos, incluído lo referente al manejo de 8 *sprites*, se realiza mediante los 47 registros del controlador de video, accediéndo a ellos mediane el comando POKE.

En el BASIC MSX, podremos encontrar numeosos comandos para la realización, creación y manejo de *sprites*, y varios modos de pantalla, cosa que no tiene ninguo de los ordenadores en cuestión.

Veamos como son las instrucciones del MSX, que permiten semejante control.

Podemos comenzar con PSET (x, y), c. Este pone el punto especificado a un color determinado. En el Spectrum, equivale a la instrucción PLOT. Luego tenemos la instrucción PRESET (x, y),c, que restituye el color de un punto. DRAW y LINE trazan líneas pero con la segunda opción se pueden dibujar cuadrados y rectangulos, con la posibilidad de poderlos colorear por dentro. Para dibujar arcos, círculos y elipses, hay un comando que lo hace directamente. CIRCLE (x, y(, r, c, ai, af, fc dibujará un círculo con las coordenadas x, y en el centro, donde r es el radio (en

radianes), c es el color, ai y af el ángulo inicial y final, respectivamente y fc es la relación existente entre el eje x e y.

El Spectrum posee esta instrucción pero con el formato algo menos compleo, CIRCLE (x, y), c, dibuja un círculo con el centro en las coordenadas x, y, con rádio c (en radianes).

También existen algunas, de estas instrucciones en el BASIC del Spectrum, aunque éste no disponga de un *chip* de video lo que empobrece la resolución gráfica, algunas de las cuales son:

— PLOT
— DRAW
— CIRCLE
— BORDER, INK, PAPER
— FLAH
— INVERSE
— OVER
— POINT
— ATTR

Como se puede comprobar, casi todos estos comandos tienen su equivalente en el MSX. Podríamos destacar, por ejemplo, que las instrucciones del Spectrum BORDER, INK, PAPER, en el MSX, equivalen a COLOR i, j, k. Sin embargo, el MSX no posee las instrucciones INVERSE, FLAHS y OVER.

El Commodore no tiene comandos directos y realiza todas las operaciones de control gráfico, tanto de creación como de manejo, mediante los consabidos POKERS a las posiciones de la memoria expresamente

Listado 6

```
1000 REM Prueba 6
1010 PRINT"P"
1020 K=0
1021 DIM M(5)
1030 DIM M(5)
1040 K=K+1
1050 A=(K/2)*3+4-5
1051 FOR L=1 TO 5
1052 NEXT L
1060 GOSUB 1120
1070 FOR L=1 TO 5
1080 NEXT L
1090 IF K<1000 THEN 1030
1100 PRINT "f"
1110 END
1120 RETURN
```

Listado 7

```
1000 REM Prueba 7
1010 PRINT"P"
1020 K=0
1021 DIM M(5)
1030 DIM M(5)
1040 K=K+1
1050 A=(K/2)*3+4-5
1051 FOR L=1 TO 5
1052 NEXT L
1060 GOSUB 1130
1070 FOR L=1 TO 5
1080 M(L)=A
1090 NEXT L
1100 IF K<1000 THEN 1030
1110 PRINT "f"
1120 END
1130 RETURN
```



preparadas para ello.

Sin embargo, en el MSX, también se puede acceder a la memoria de video y controlar desde allíi la creación de gráficos y *sprites*. Esto se hace gracias a instrucciones expresamente creadas para ello.
VPOKE y VPEEK, son dos ejemplos claros de acceso directo a cualquier parte de los 16 K de memoria video que tienen todos los MSX.

No todo está escrito sobre los comandos gráficos del MSX, ya que todavía faltan los subcomandos que utlizados con la instrución DRAW, permiten dibujar en cualquier sentido y en cualquier posición.

En realidad, este punto es el más fuerte de los MSX haciéndolos insuperables en este terreno, aunque también tiene otro frente bastante importante y muy completo a la hora de programar; la creación de sonidos.

Instrucciones de sonido

Viendo las características del MSX en el apartado de gráficos, uno podría pensar que aquí se acaba la historia, cuando en realidad es su inicio.

Siendo el único que, junto con el Commodore 64, posee un *chip* para la generación de sonido. Ambos *chip* son polifónicos (3 voces), disponiendo el MSX de 8 octavas, mientras que el Commodore tiene 9. Pero de nuevo, la potencia del BASIC MSX, se hace notar. ¿Resultado?

Crear sonidos en el MSX es taréa

TABLA COMPARATIVA ENTRE LOS ORDENADORES MAS EXTENDIDOS

	MSX	SPECTRUM	COMMODORE
LISTADO 1	2.23	4.73	1.60
LISTADO 2	6.01	8.57	9.80
LISTADO 3	17.45	20.98	18.40
LISTADO 4	18.54	20.15	20.30
LISTADO 5	19.75	23.93	22.01
LISTADO 6	32.12	54.29	32.61
LISTADO 7	45.15	78.91	51.22
LISTADO 8	1.41	1.37	0.90



FOTO 4:
ZX Spectrum, una filosofía aplicada por Sir Clive Sinclair al mundo del ordenador personal.

FOTO 5:
Vista posterior del Commodore 64. Las conexiones no son estándar, por lo que es necesario utilizar periféricos destinados solamente para este ordenador.



bastante más sencilla de lo que uno puede imaginar, mientrar que el Commodore 64, obligará al usuario a estudiar los contenidos de la memoria y a usar innumerables instrucciones POKE, que es la única manera de controlar el chip de sonido.

El Spectrum no tiene tal *chip*, sólo cuenta con una instrucción BEEP con dos parámetros que controlan la nota y la duración, esto hace prácticamente inexistente la posibilidad de generar sonidos en el Spectrum.

Veamos el comportamiento de estas instrucciones.

BEEP, genera en el MSX un leve chasquido, mientras que en el Spectrum, utilizando las coordenadas necesarias, como la duración y la nota, puede llegar a crear algo de sonido.

MSX, Commodore 64 y ZX Spectrum, tres ordenadores, tres visiones distintas del ordenador personal.

PLAY, se usa para tocar música. Ve precedida por una serie de subcomandos que indican las notas a tocar las caracteristícas del sonido, como pueden ser ritmo, volumen y la octava.

Los subcomandos usados en esta instrucción son:

C = DO D = RE E = MI F=FA G=SOL
A = LA B = SI

V— va precedido de un número entre 0 y 15, sirve para especificar el volumen de la nota.

O— va precedido de un número entre 0 y 8, especifica la octava de las notas a tocar.

R— genera una pausa en la generación de sonidos.

T— seguido de un número especifica la velocidad de efecución de la secuencia de notas.

S— especifica la forma de la onda.

M— especifica el período de envolvente.

De cualquier manera, para completar este apartado resaltaríamos la necesidad de leer el artículo publica-

ZX y Commodore abriéron las puertas y los ojos de aquellos que implantaron y vieron la necesidad de establecer una estandarización en los ordenadores personales.

do en el número 2 de MSX, denominado Generación de Sonidos, donde se explica con todo lujo de detalles, las innumerables posibilidades existentes en la realización de sonidos.

Acabando el tema podemos ver una de las características más importantes del MSX; la posibilidad de manejar interrupciones, algo típico en ordenadores de más nivel.

Manejo de interruptores

El MSX dispoe de una serie de instrucciones para el manejo y control de interruptores; éstas se pueden producir de múltiples maneras:

1. Interrupciones por error.

Se producen cuando se genera un error en el programa: normalmente devuelven el control al intérprete BASIC pero se pueden controlar mediante una subrutina habilitada para tal fin dentro del mismo programa. Este se puede realizar con la instrucción:

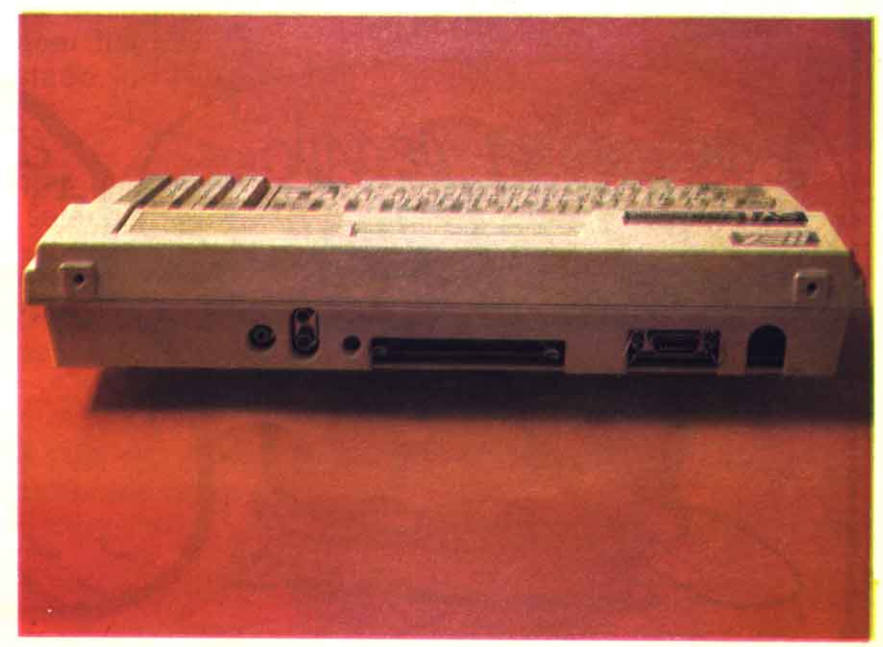
IN ERRO GOTO xxxx, que producirá un salto a la línea xxxx que deberá finalizar con la instrucción RESUME.

```
Listado 8

1000 REM Prueba 8
1010 PRINT"P"
1020 K=0
1021 DIM M(5)
1030 DIM M(5)
1040 K=K^2
1050 B=LOG (K)
1051 FOR L=1 TO 5
1052 NEXT L
1060 C=SIN(K)
1070 IF K<1000 THEN 1030
1080 PRINT "f"
1090 END
```

FOTO 6: Vista posterior ZX Spectrum. Los interfaces son una constante en el desarrollo de aparatos y periféricos para el ordenador.


Foto 7: Conectores universales que potencian un standard.



2. Interrupciones producidas por las teclas de función.

Se produce cuando se pulsa una de las teclas de función. El control de estas interrupciones se realiza mediante la intrucción:
ON KEY GOSUB x1, x2, x3, ..., que saltará a la subrutina en función de la tecla que se hala pulsando. Esta interrupción se puede habilitar o no con el comando KEY ON/OFF.

3. Interrupciones por la pulsación de la barra espaciadora o alguno de los disparadores de un *joysticks*.

Se controlan mediante la instrucción:
ON STRIG GOSUB y se habilita o no con el comando STRIG ON/OFF.

4. Interrupciones por la pulsación simultánea de las teclas CTRL y STOP.

Estas se controlan mediante la instrucción ON STOP GOSUB, se habilita mediante el comando STOP ON. Es muy útil para evitar la detención de un programa y de esa forma protegerlo.

5. Interrupciones del temporizador.

Son las que se producen cada cierto tiempo. Estas entran en funcionamiento mediante la instrucción:
ON INTERVAL = tiempo GOSUB y se habilitan mediante el comando INTERVAL ON.

En la realización del potente BASIC del MSX, se nota la mano de una importante compañía: Microsoft.

Conclusiones

Varios son los puntos a destacar de esta pequeña comparativa. En primer lugar, se realizó en base al ordenador solamente y al lenguaje que éste posee sale de fábrica. Como mencionábamos anteriormente, debido a que el ZX Spectrum y el Commodore, llevan más tiempo en el mercado, indudablemente tienen el *software* más diverso y completo que hay en la actualidad, hay desde el típico juego de marcianos hasta una aplicación del BASIC, sin contar con los diversos lenguajes de programación que cuentan, además del BASIC.

Segundo, la creciente ola de periféricos y la mejora en que se han visto las diversas casas comerciales al aparecer el MSX con todos los alicientes de un ordenador grande, ha obligado a crear desde unidades de *diskettes* hasta controladores domésticos.

Ninguno de éstos, ha sido objeto de estudio y creemos conveniente, por esta razón, continuar en algún número futuro la comparativa entre estos ordenadores ampliados hasta el límite de sus posibilidades.

Para finalizar, destacar que tampoco se ha entrado en características de los distintos lenguajes, debido a que se podrían escribir ríos sobre todos y cada uno de ellos. Esperemos, pues, que estas pequeñas indicaciones hallan servido para asentar y diferenciar las características esenciales del lenguaje de programación de cada ordenador. No hemos entrado ni en las interioridades de cada uno, ni en los mapas de memoria, por considerarlo ajeno a lo que deseábamos explicar.

De cualquier manera, hemos realizado una prueba para comprobar la velocidad de ejecución de los distintos BASIC, y aunque sólo sea orientativo, el cuadro muestra los tiempos obtenidos en la ejecución de los diferentes programas. Estos también se exponen, por si algún lector desea realizar la comparación con amigos que tenga cualquier otro ordenador personal.





GANE
7.000 PTAS.
todos los meses
PARTICIPANDO
EN NUESTRO
CONCURSO
A partir del próximo número MSX premiará mensualmente los programas que hagan llegar los lectores.
Para participar en este concurso abierto, todo aficionado a los ordenadores con este estándar deberá hacer llegar a la redacción de la revista el listado, un cassette y un texto explicativo.
Entre todos los programas que recibamos cada mes, serán seleccionados para su publicación aquellos que reúnan los siguientes criterios:
— Originalidad de la aplicación.
— Simplicidad del método de programación.
La única condición para participar en el concurso será que los programas no hayan sido publicados previamente en ninguna revista.

Aunque nunca se haya parado a pensar en ello, en muchos programas habrá visto figuras (naves, coches, pájaros, bombas,...) que se mueven en la pantalla, que pueden pararse, acelerar, desaparecer, que combinándolas se consigue que haya acción en la pantalla, que haya una auténtica película de dibujos animados.

Estas figuras no se trazan con las órdenes de gráficos (CIRCLE, DRAW, LINE, PSET Y PRESEY). Funcionan independientemente de los comandos gráficos, y tanto es así, que también pude haber figuras móviles en el modo de textos SCREEN 1, donde no puedes utilizar esas cinco órdenes. Estas figuras son los *"sprites"*, y vamos a contaros todo sobre ellas.

¿Qué es un *Sprite*?

Un *sprite* es un figura móvil monocroma que se puede desplazar por la pantalla por encima de todo lo que haya escrito o dibujado. Nada puede aparecer escrito o dibujado sobre un *sprite*, más que otro *sprite*. Vamos a examinar esto más detenidamente.

Lo que vea en su televisor o en su monitor como la pantalla del ordenador, no es más que la superposición de 33 planos de dibjos distintos. El último de esos planos paralelos, el que se halla más al fondo de la pantalla, es el plano principal, el plano donde se desarrollan los textos y los gráficos. Los otros 32 planos de dibujos se hallan tapando a este plano principal, y su función es exclusivasmente dibujar figuras móviles. Están



SPRITES MSX:

DIBUJOS ANIMADOS EN TU ORDENADOR

Una de las muchas ventajas que posee la norma MSX en comparación con otros ordenadores es la de disponer de figuras móviles, de "sprites". En este artículo explicaremos como poder crear y mover sus propios Sprites desde el BASIC MSX

numerados del 0 a 31, siendo el plano número 0 el más lejano al plano principal, y el más cercano al observador, y el plano número 31, el plano contíguo al plano principal, y el más lejano al observador. Por eso, una figura móvil dibujada en el plano 0 tapará todas las figuras móviles y todos los dibujos y letras que hubiera en los *píxels* que ella ocupa, pues este plano está tapando a todos los demás. Una figura móvil dibujada en el plano 1 tapará todo lo que ocupe sus mismos *píxels* y éste dibujado en planos inferiores, como el 2, el 3,...hasta el plano 31 y también el plano principal, pero podrá ser tapada a su vez por una figura móvil del plano 0. Las figuras móviles del plano 2 podrán ser tapadas por las de los planos 0 y 1, pero taparán a su vez a las de los planos 3, 4, 5,... 31 y plano principal, y en esta sucesión podríamos seguir hasta el plano 31, cuyas figuras móviles pueden ser tapadas por todas las figuras móviles de los planos 0 al 30, pero pueden tapar los gráficos y letras del plano principal, los cuales, al estar en el útimo plano, pueden ser tapados por todas las figuras móviles que se dibujan en los planos 0 al 31.

Tipos de *Sprites*

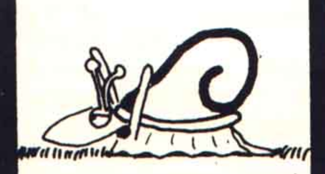
No todas las figuras móviles pueden ocupar la misma cantidad de *píxels*. Según esta cantidad, podemos distinguir cuatro tipos de figuras móviles:

TIPO 0: figuras móviles de 8 filas × 8 columnas de *píxels*
TIPO 1: figuras móviles de 8 filas × 8 columnas de "*píxels* ampliados"
TIPO 2: figuras móviles de 16 filas × 16 columnas de *píxels*
TIPO 3: figuras móviles de 16 filas × 16 columnas de "*píxels* ampliados"

Estos tipos son imcompatibles, es decir, no pueden salir a la pantalla figuras móviles de un tipo y de otro tipo al mismo tiempo. Para acceder a un tipo concreto de figura móvil,se utiliza el segundo dato de la orden SCREEN. Por ejemplo si queremos obtener *sprites* de tipo 2 en SCREEN 1, habrá que teclear SCREEN 1, 2.

Los "*píxels* ampliados" son bloques de 4 *píxels* (2 filas × 2 columnas) que en el tipo 1 y en el tipo 3 se






utilizan como si fueran *píxels*. Por lo tanto, para un figura móvil del tipo1 se suministran los datos como si fuera de 8 × 8 *píxels* aunque en realidad sea de 16 × 16, y en realidad es de 32 x 32.

Una aplicación de estos *"píxels* ampliados" puede ser, por ejemplo, cuando tenga que hacer una explosión, hacerla primero en el tipo 0, después en el tipo 2 y por último en el tipo 3 (con *"píxels* ampliados") con los cual se conseguirá la impresión de que la nube de la explosión ha ido aumentando. Esto no es otra cosa que una más de la inmensa cantidad de aplicaciones de los *"píxels* ampliados" que ya irá descubriendo a medida que practique con sus figuras móviles.

Definiendo *Sprites*

Vamos a definir una figura móvil; por ejemplo, una gaviota. Antes de nada habrá que diseñarla (figura 1). Hemos elegido el tipo 0 de figuras móviles de 8 × 8 *píxels*, pues es el más sencillo de elaborar.

En el diseño de una figura móvil es muy importante recordar queaquel color del que vayamos a pintar esta figura móvil (sólo uno, los *sprites* son monocromos) sólo afectará a los *píxels* que hayamos decidido que constituyen la figura móvil, pues el resto de los píxels que complen las 8 filas x 8 columnas, y que aparecen en blanco en la figura 1, en realidad las letras o dibujos que se desarrollen en estos 8 × 8 *píxels* y cuyos píxels coincidan con los *píxels* coloreados de la figura móvil aparecerán del color de la figura móvil, pero aquellos dibujos o letras que estén dentro de los 8 x 8 *píxels* y cuyos *píxels* coincidan con los definidos como transparentes en la figura móvil aparecerán perfectamente.

Esto se cumple también en cualquier otro tipo de figuras móviles.

Siguiendo con el diseño de la figura 1, ahora correspondería pasar las filas horizontales de 8 *píxels* al sistema binario, considerando los *píxels* coloreados como "1" y los transparentes como "0". A continuación, convienen pasar estos datos al sistema numérico hexadecimal, o de base 16:

```
& B00000000=&H0
& B00000000=&H0
& B00100010=&H22
& B01010101=&H55
& B10001000=&H88
& B00001000=&H8
& B00000000=&H0
& B00000000=&H0
```

Para definir un *sprite* desde BASIC, hay que utilizar el mando SPRITE$. Tras la palabra SPRIT$ hay que indicar entre paréntesis el número de figura móvil a que se refiere. Como el ordenador puede acordarse de hasta



256 figuras móviles) del tipo 0 y del tipo 1) las numera de 0 al 255. Así, si numeras esta figura móvil como la figura número 0, tendrás que poner:

```
SPRITE$ (0) = CHR$ (&H0) + CHR$
(&H0) + CHR$ +
+ CHR$ (&H55) + CHR$ (&H88) +
CHR$ (&H0) + CHR$ + CHR$ (&H0)
```

Tras el signo "igual", como ya has visto, tendrás que poner ocho veces la función "CHR$", y entre paréntesisis y por órden de arriba a abajo, los datos en hexadecimal de las líneas horizontales de tu figura móvil (el color se detemina después con otra orden).

Pero de esta manera sólo se pueden definir las figuras móviles del tipo 0 y del tipo 1. Para definir las figuras móviles de 16 filas × 16 columnas, ya sea de *píxels* o de *píxels* ampliado, hay que dividir esta figura móvil en cuanto "subfiguras móviles", de 8 filas × 8 columnas, e introducir los datos de estas subfiguras según el orden indicado en la figura 2. Para esta operación necesitarás usar 32 veces la función CHR$, para introducir los datos hexadecimales de las líneas horizontales de 8 *píxels*.

Has de tener en cuenta también que la figuras móviles del tipo 2 y del tipo 3 ocupan 4 veces más en la memoria que las del tipo 0 y del tipo 1, luego el ordenador no puede guardar en su memoia más de 64 giguras móviles de 16 filas × 16 columnas, y habrá que numerarlas del 0 al 63.

Para que los mandos SPRITE$ no resulten tan largos, hay varios trucos que se pueden emplear. Puede hacer en otras órdenes, por ejemplo, que la varible A$ sea igual a la suma de las ocho funciones CHR$ de la primera subfigura, que B$ tenga las ocho de la segunda, C$ de la tercera y D$ de la cuarta, y así podrás teclear, como *sprite* de las 16 filas × 16 columnas con el número 27, por ejemplo:

```
SPRITE$ (27) = A$ + B$ + C$ + D$
```

y tu figura móvil quedará definida.

También puedes sustituir CHR$ por los caracteres entrecomillados cuyo código en hexadecimal sea uno de los datos. Por ejemplo, al definir la gaviota, si sabemos que el carácter número 55 hexadecimal, es decir el número en decimal e la "U" mayúscula, podemos teclear:

```
SPRITE$ (0) = CHR$ (&H0) + CHR$
(&H0) + (&H22) + "U" + CHR$
(&H88) + + CHR$ (&H8) + CHR$
(&H0) + CHR$ (&H0)
```

y la figura móvil definidaseguirá siendo la gaviota, exactamente igual.

Al utilizar la segunda coordenada de la orden SCREEN, te aconsejamos bastante prudencia, pues al seleccionar el tipo de figura móvil que



quieres utilizar, al mismo tiempo borra de la memoria absolutamente todas las figuras móviles que hubieras definido previamente, aunque fueran del mismo tipo que has selccionado. Esto no sucede, sin embargo, con la primera coordenada de la orden SCREEN, por lo que puedes cambiar de modo de pantalla (de texto a gráfico, de gráficos a texto, de gráficos entre sí, e incluso llegar a SCREEN 0, el modo de texto en que no se pueden usar figuras móviles) sin temor a que tus figuras móviles se borren de la memoria.

Movimiento Sprites

Para situar tus figuras móviles en la pantalla y moverlas a continuación, es conveniente que sepas algo más sobre los planos de pantalla de los que hablábamos antes.

El plano de dibujo principal se divide en 192 filas × 256 columnas de *píxels*, numeradas, del 0 al 191 y del 0 a 255, apareciendo este plano íntegro en la pantalla, pero no sucede lo mismo con los planos de figura móvil. Los planos de figura móvil se dividen en 256 filas ×256 columnas de *píxels*, numeradas ambas del 0 al 255. Al caber sólo en la pantalla 192 filas de *píxels*, hay una parte de los planos de la figura móvil que no aparecen en la pantalla y que va desde la fila 192 hasta la fila 255 de *píxels*. Esta zona se halla oculta en el borde inferior de la pantalla.

Como los *píxels* de los planos de figura móvil se hallan exactamente en el mismo lugar que los *píxels* de plano principal (por ejemplo, el píxels (27, 49) de cualquier plano de dibujo de figuras móviles se superpone a píxels (27, 49) en cualquier plano inferior, incluido el plano prinipal. Vamos, pues, a situar una figura móvil.

Si la gaviota que definimos en el apartado anterior se mantienen en lamemoria de tu ordenador (si no vuélvela a definir) podemos situarla en la pantalla. Antes de situarla, tendremos que decidir en qué plano la vamos a poner, en que *píxels* y de qué color. Decidimos al azar que la situamos en el plano número 3, en el cuadrado de *píxels* con vértices (100, 100), (107, 100), (107, 107) y de color blanco (15). Para qe se vea, pues de color blanco, hay que hacer que el color del fondo sea distinto color. Azul, por ejemplo. Por lo tanto pon tu ordenador en SCREEN 1 y teclea:

```
10 color, 4, 4
20 put sprite 3, (100, 100), 15, 0
```

La línea 10 establece que el color del fondo y de los bordes sea azul, y a línea 20, utilizando el mando "PUT SPRITE", sitúa la figura móvil tal y como habíamos decidido. Tras un mandato "PUT SPRITE", hay que indicar el plano donde se sitúa la figura móvil, el vértice superior izquierdo del cuadrado donde se sitúa, el color, y la figura móvil de que se trate (la gaviota la numeramos con el sprite número 0).

Ahora te encontrarás con que tienes una gaviota en medio de la pantalla y no se puede borrar, ni con "BS", ni con "DELETE", ni con la orden "CLS": Pero sí se puede quitar de dos maneras: poniéndola transparente (color 0):

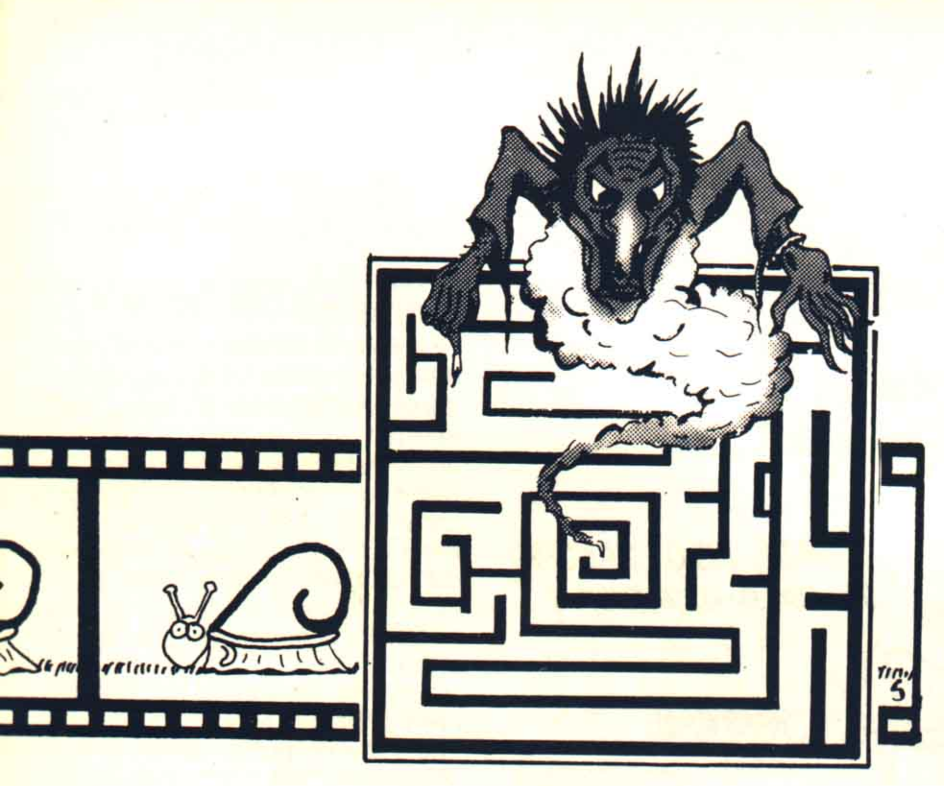
```
PUT SPRITE 3, (100, 100), 0,
0,
```

o poniendo otra gaviota en el mismo plano de la figura móvil, pero en la zona que no aparece en la pantalla, como el cuadrado de vértice superior (200,0):

```
PUT SPRITE 3 (200,0), 15, 0
```

Te estarás preguntando por qué el hecho de poner una gaviota fuera de la pantalla ha hecho borrarse a la anterior, cuando lo lógico sería que permaneciesen dibujadas ambas, una que no puede verse, fuera de la pantalla, y otra, la de antes en medio de la pantalla. La respuesta es fácil, en un plano de figura móvil, solo cabe **una** figura móvil La que había en el plano 3 era una gaviota en medio de la pantalla, pero al decirle al ordenador que me situará otra en el mismo plano, pero fuera de la vista, lo ha hecho, pero ha borrado a la gáviota anterior, para tener sólo una figura móvil.

Esta caracterítica de los planos de figuras móviles limita la cantidad de figuras móviles que puede aparecer en un instante de tiempo a 32, una por cada plano de figura móvil, aunque haya definidas más de 32 figu-



ras móviles en la memoria del ordenador. Esto, lejos de ser una desventaja, resulta muy beneficiosos para crear el efecto de los dibujos animados. Vamos a poner un ejemplo. La gaviota de antes va a ir volando de un lado a otro de la pantalla. Para ello necesitamos las dos posiciones de la gaviota: la gaviota con las alas levantadas y la gaviota con las alas bajadas. La primera posición ya está definida y la segunda posición es la figura 3. Definimos, antes de nada, la segunda posición, por ejemplo, como la figura móvil número 1:

```
& B00000000=&H0
& B00000000=&H0
& B00000000=&H0
& B00010100=&H14
& B00101010=&H2A
& B01001001=&H49
& B10000000=&H80
& B00000000=&H0
```

luego el programa será, suponiendo que siga la otra gaviota en la memoria el siguiente:

```
10 SPRITES$ (1) = CHR$ (&H0) + CHRS$ (&H0) + CHR$ (&H14) + CHR$ (&H2A) ++CHR$ (&H49) + (&H80) + (&H0)
20 color, 4, 4: SCREEN 2
30 FORT = 0 TO 100 PUT SPRITE 3, (100,00), 15, TMOD 2:
FORL = IT050: NEXT: NEXT
```

Este programa solo hace batir las alas a la gaviota, sin moverla. Para ello pone la figura móvil a instalar en función del resto de la división de la variable del bucle T entre 2. Hemos incluido un buce L ralentizador del movimiento de la gaviota.

Vamos ahora a moverla por la pantalla. Cambia la orden 30 por la siguiente:

```
30 FOR T=1TO 200: PUT SPRITE 3, (T, 100), 15, T MOD 2: FOR L = 1TO 50: NEXT: NEXT
```

y así verás que ti gavióta se desplaza por la pantalla desde el punto (0, 0) al (0, 200).

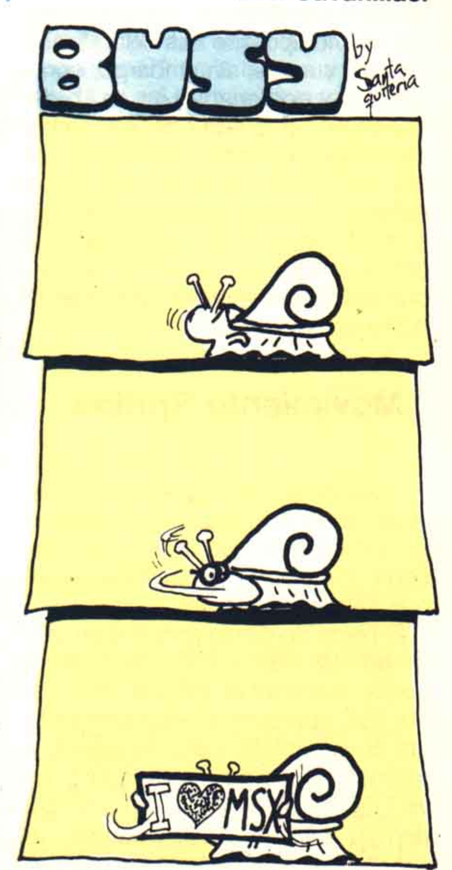
Te vamos a explicar ahora un fenómeno que sucede con las figuras móviles, y que en inglés se llama *"wrap–Around"*. Cambia el bucle T para que, en vez de ser de 1 a 200, sea de 1 a 1000 (FORT = 1 TOL1000) cuando el bucle llega a valo res como 300, 400.etc...,la gaviota sigue aún viéndose. Esto sucede porque al llegar al *píxels* (256,100) el ordenador sitúa a ls figuras móviles en el (0,100) y parece como si la gaviota comenzará de nuevo a atravesar la pantalla. Esto también sucede en vertical. Cambia la orden 30 por:

```
30 FORT = 0 TO 1000: PUT SPRITE 3, (100, T) 15, TMOD 2: FOR L = 1TO50: NEXT: NEXT
```

y sucederá algo parecido. La diferencia entre el *"Wrop–Around"* vertical y el horizontal es que en el vertical la figura móvil tiene que atravesar la zona oculta de su plano antes de llegar al punto (100, 256) y a causa de esto situarse en el punto (100,0).

Por último te diremos que hay un solo inconveniente en la salida a pantalla de los *sprites* MSX: no puede hacer más de cuatro figuras móviles en una misma línea horizontal. Si se sitúan 5, sólo se podrán ver las cuatro que se hallen en los cuatro planos de figura móvil. Así si se hallan figuras móviles de los planos 0, 2, 4, 5, 7, 9, y 11 en la misma línea horizontal, solo verán visibles las de los planos 0, 2, 4, y 5. Si una figura móvil se sitúa alineada horizontalmente con cuatro o más figuras móviles de planos superiores, no se verá, pero si se halla al menos 1 *píxels* por debajo de los demás será visible solo en parte, de las rebanadas de *píxels* que se hallen desplazadas de las demás.

J.M. Cavanillas.



SVI-728 MSX-PLUS.

NO HAY QUIEN DE MAS: UN EQUIPO EXCEPCIONAL, A UN PRECIO DE EXCEPCION.

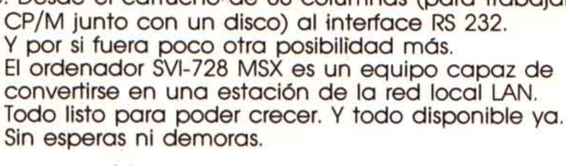
TODOS DE ACUERDO. "MSX": STANDARD MUNDIAL



COMPATIBLE

Este sería el primer adjetivo a destacar del ordenador SVI-728 MSX: su total compatibilidad con el sistema standard mundial MSX.

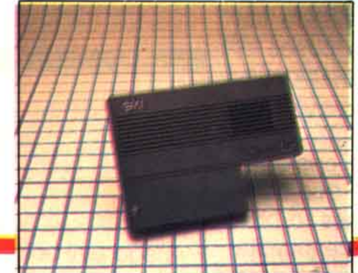
ILIMITADO



Gracias a la amplia variedad de periféricos disponibles (y todos compatibles MSX) el ordenador SVI-728 no tiene límites. Desde la unidad de disco al cassette. Desde el cartucho de 80 columnas (para trabajar en CP/M junto con un disco) al interface RS 232. Y por si fuera poco otra posibilidad más. El ordenador SVI-728 MSX es un equipo capaz de convertirse en una estación de la red local LAN. Todo listo para poder crecer. Y todo disponible ya. Sin esperas ni demoras.



Infórmate en tu Concesionario Autorizado Spectravideo y comprueba la total "modularidad" del sistema SVI-728. Nacido para crecer.



Spectravideo.
La informática del futuro, hoy.







Algebra de BO

Antes de entrar en definiciones y explicaciones, que en lugar de facilitar la lectura, la complicarían, iniciemos este artículo con una breve historia del padre del álgebra que lleva su apellido. George Boole, nacido en 1815, será, sin lugar a dudas recordado por todos los que, de una forma u otra, están ligados a la profesión informática. gracias a sus teorías y a sus libros, ha sido posible entender la complicada lógica aristotélica. Desarrolló una serie de símbolos, que utilizados en combinación con otros elementos, obtenían una serie de resultados. Estos elementos y símbolos, los veremos con las instrucciones que posee el ordenador.

Es interesante saber que esta rama matemática, es necesaria entender a la perfección, ya que la construcción de instrucciones con el formato IF–THEN–ELSE, se verá beneficiado o perjudicado, en base a nuestro conocimiento y saber hacer.

Las tablas de verdad son otra forma de aplicar estos teoremas y estas leyes. Pero empecemos viendo las distintas instrucciones que posee el MSX y que se utilizan con más o menos frecuencia. Existen seis operadores lógicos, que son NOT, AND, OR, WOR, EQV e IMP. Los operadores lógicos son elementos, que colocados entre dos condiciones (o proposiciones) permiten una respuesta del tipo TRUE o FALSE (verdadero falso, 1 ó 0, etc.).

Este tipo de respuesta es debido a que el ordenador está formado por un conjunto de interrupciones que sólo pueden estar encendidos o apagados. Una combinación de interruptores es lo que hemos denominado en anteriores artículos como *bytes* (8 interruptores). La lógica del ordenador, por lo menos, lo que hace que la máquina razone, es este conjunto de reglas que conforman el algebra de Boole.

Una vez entendido que el mecanismo interno de la máquina se rige por impulsos eléctricos y por unas determinadas reglas, vamos a explicar los distintos operadores lógicos y sus tablas de verdad.

En orden de importancia, podemos destacar los tres operadores más comocidos por todos los lectores. Estos son el OR, AND y NOT (que significa O, Y y negación), aunque el MSX posee otros tres, XOR, IMP y EQV, que se pueden expresar como combinaciones de los tres primeros.

En primer lugar, explicaremos los diversos elementos que componen este Algebra individualmente y sus características principales, dejando los casos prácticos para el final. Es importante entender el buen funcionamiento de estos elementos, que además nos permitirán realizar auténticas diablуras a la hora de complicar las sentencias condicionales IF–THEN–ELSE. Iniciemos el estudio de la tabla por el caso más sencillo, se trata del operador NOT.

Este es la negación de cualquier elemento que venga a continuación, su tabla es;

X	NOT X
0	1
1	0

Como podemos observar, con una condición (X) sólo pueden ocurrir dos casos, que sea cierta (1) o falsa (0), por lo que al aplicar este operador, obtendremos como resultado el contrario del valor inicial.

Continuemos con el operador siguiente en orden de importancia y que nuestros lectores conocerán de haberlo visto en las instrucciones IF.

Permite comparar dos condiciones y ejecutar una acción, caso de que el resultado sea cierto. Como es obvio, se necesitan dos argumentos o condiciones que denominaremos X e Y, que pueden tomar hasta un total de cuatro valores que :

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

Según la tabla, una instrucción del tipo X AND Y, sólo será cierta cuando las condiciones sean ciertas, es decir, "si ocurre la condición X" y "la condición Y".

Bien, es hora de introducir un ejemplo que ayudará a aclarar estos conceptos.

En los programas, a veces nos encontramos con instrucciones IF con el siguiente formato:

```
810 IF RC  3 AND CC     1
THEN 1040
```

Si la destripamos un poco y aplicamos lo visto hasta el momento, podemos darnos cuenta de que el programa SOLO ejecutará la línea 1040 en el caso de que RC sea mayor que 3 (condición X) y que CC sea distinto de 1 (condición Y). Es decir, que si observamos la tabla, equivaldría al último caso. De nada sirve que se

De la mano de George Boole, se inició una rama de la matemática imprescindible para el buen desarrollo de aplicaciones.



Elemento fundamental en el desarrollo de aplicaciones y ayuda primordial en la toma de decisiones, el Algebra de Boole, se convierte, con el MSX en un protagonista digno de tener en cuenta. Sobre todo, con la posibilidad de poder implementar cualquier función de este álgebra, gracias a una serie de instrucciones que comentaremos a continuación.

cumpla la primera condición y la segunda no, o que se cumpla la segunda pero no la primera, menos aún si no se cumple alguna de las dos. Comprendido esto, pasemos al último caso de los tres lógicos, OR. Este al igual que el anterior, compara entre entre dos condiciones, la posibilidad de que una (X) y otra (Y) condición sea cierta. Su tabla es:

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

Si la comparamos con el caso anterior, observamos tres posibilidades en que la condición final sea cierta. Aquí sólo basta que una de las dos condiciones sea verdadera para quién el resultado también lo sea. Veamos el ejemplo anterior y sustituyamos el operador lógico AND por OR.

```
810 IF RC  3 OR CC   1
THEN 1040
```

En este caso, para ejecutar la línea 1040 basta con que una de las condiciones sea verdadera, que RC sea mayor que 3 o que CC sea distinto de 1.

Antes de explicar el siguiente operador, volvamos a ver las tablas. Quiero hacer constar que estos elementos del Algebra de Boole, tienen sus honónimos en otras operaciones. ¿Por qué no sustituís el operador lógico OR por el signo '+' y el operador lógico AD por el signo '×' y desarrolladas las tablas de nuevo? ¿Qué resultado obtendreís al efectuar el cambio?

Las respuestas a estas dos preguntas las dejamos a manos de los lectores, de esta forma podreís convenceros de que todas las ramas de las matemáticas se interelacionan.

Pues bien, hasta aquí hemos visto los tres operadores más importantes y más populares, en los que a programación se refiere. Los vemos continuamente en los programas y rutinas. Los que vamos a estudiar a continuación, no son tan extendidos ni tan conocidos como las anteriores, pero, son tres elementos que caracterizan y potencian, a la vez que elevan el nivel, de nuestro ordenador MSX.

Estos tres operadores lógicos son: XOR, IMP y EQV. La primera se denomina OR–exclusiva, la segunda es la implementación y la rutina es la función de equivalencia.

Vamos a realizar un pequeño paréntesis a estas alturas, para reiterar al lector, y perdonar la insisitencia, que estos operadores ayudarán a potenciar la construcción de la sentencia IF–THEN–ELSE. Más adelante y para cerrar este artículo, veremos unos ejmplos y explicaremos algunos propiedades del Algebra de Boole.

Las tablas de verdad, son otra forma de aplicar los teoremas y las leyes de Algebra de Boole.

Continuando con estos tres operadores, iniciemos esta segunda etapa con el XOR.

La función que desarrolla este operador es la misma que si utilizamos los operadores AND, OR y NOT. Es decir, que si realizamos la tabla de verdad de la función X XOR Y, obtendremos como resultado una tabla que será igual a la que se obtienen haciendo la tabla de: (X AND (NOT Y)) OR ((NOT X) AND Y). Esto podrá sonar a Chino, pero si sustituímos las condiciones X e Y, por valores auténticos, es decir, funciones del tipo: A 132 o incluso C$ "pepe", podremos comprobar las inmensas posibilidades de este operador.

Por ejemplo, podemos tener una instrucción del tipo:

```
IF (A 132 AND (NOT
c$ "pepe")) OR ((NOT
a 132) AND c$ "pepe")
THEN GO TO 1399
```

y mediante la aplicación de este operador reducirla a:

```
IF a 132 XOR c$ "pepe"
THEN GO TO 1399
```

Esto es posible gracias a la aplicación de la tabla de esta función, que es:

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

Con esta tabla y las que hemos visto anteriormente, los lectores podrán realizar las tablas de las dos funciones expuestas anteriormente y comprobar su igualdad.

A continuación, hablaremos de la función de equivalencia, que en el BASIC MSX se representa por EQV. En esta, al contrario que la anterior, tiene la particularidad de tomar valores verdaderos (1) cuando las dos condiciones son ciertas, es decir, cuando X e Y sean iguales a 1. Podemos deducir, que su tabla de verdad será:

X	Y	3 EQV Y

Algebra de Boole

0	0	1
0	1	0
1	0	0
1	1	1

Por último, queda por comentar el operador, IMP. No tiene muchas aplicaciones, por lo que se verá pocas veces. De cualquier manera, esto no es razón para no comentarlo.

La función IMP, es la implementación. Esta equivale a hallar la tabla de verdad de la negación del primer operador y a continuación, realizar la función OR entre el resultado.

Su resultado es la siguiente:

X	Y	X IMP Y
0	0	1
0	1	1
1	0	0
1	1	1

Pues bien, aquí acaba una introducción de los operadores lógicos. Hagamos ahora, un sumario de las tablas antes de entrar en más materia.

Sumario de los operadores lógicos del BASIC MSX

X	Y	AND	OR	XOR	EQV	IMP
0	0	0	0	0	1	1
0	1	0	1	1	0	1
1	0	0	1	1	0	0
1	1	1	1	0	1	1

Ya hemos visto todos los operadores lógicos que posee el BASIC MSX, pero la explicación se queda imcompleta si no vemos algunas relaciones importantes que nos podremos encontrar en los programas, y como no, poderlas realizar en los nuestros. Todas estas relaciones se pueden verificar mediante las tablas de verdad, de las que tanto hemos hablado en este artículo.

En total son 7 casos que podemos utilizar y que simplificarán el desarrollo de los programas. Empezaremos con el operador OR.

1– X OR O = X
2– X OR X = X
3– X OR X OR X OR... = X
4– X OR (NOT X) = 1
5– X OR 1 = 1
6– X OR (X AND Y) = X
7– X OR ((NOT X) AND Y) = X OR Y

Las relaciones con el operador AND, son similares y son las siguientes:

1– X AND 0 = 0
2– X AND X = X
3– X AND X AND X AND... = X
4– X AND (NOT X) = 0
5– X AND 1 = X
6– X AND (X AND Y) = X AND Y
7– X AND ((NOT X) AND Y) = 0

todas las relaciones se verifican por el procedimiento explicado anteriormente. Si queremos hacerlo por nuestra cuenta, no olvidar que tanto X como Y, son expresiones matemáticas o lógicas. Estas también pueden ser números solamente, por lo que, para empezar a comprender las tablas haremos el siguiente ejercicio.

Tomemos dos valores numéricos totalmente aleatorios, que pueden ser el 145 y el 58. Ahora comprobemos los catorce casos que se han explicado anteriormente, siendo la X el valor 145 y la Y el 58. A continuación y sustituyendo dichos valores en las expresiones comprobemos los resultados. Haremos 5 ejemplos y dejaremos al lector la ejecución del resto hasta los catorce que lo componen.

El comando PRINT nos ayudará en los ejemplos.

Estos son algunas de las operaciones posibles. Siguiendo la pauta de la revista, queremos que el lector sea partícipe directo de la lectura y desarrollo de los artículos. Por esta razón, dejamos en sus manos completar la totalidad de los ejemplos.

De la misma forma que hemos tomado valores numéricos en las expresiones X e Y, también podríamos haber puesto cualquier otra expresión, del tipo a$="alfa" y"b =15. Este ejemplo está realizado con números puesto que es más fácil de explicar y entender.

Para acabar, veremos como funcionan estos operadores dentro del contexto de la instrucción IF.

Estudiar el funcionamiento de esta instrucción no es el objeto de este artículo, por lo que nos profundizaremos en el tema.

La sintaxis de esta instrucción es:
IF (condición) THEN (acción)

En este caso, la condición puede ser cualquier combinación de números, letras o números y letras. En estas condiciones se usan los operadores aritméticos, que estaremos acostumbrados a ver y son: , , =, , = y =. Dentro de estas condiciones podemos utilizar también funciones matemáticas del coseno, seno, arctg, etc. El resultado de estas condiciones, puede ser cualquiera, como corresponde a las comparaciones. Pero existen otras comparaciones que se efectuan con

Los operadores lógicos, AND, OR y NOT, junto con los números binarios, 0 y 1, forman la base de toda la circuitería interna de un ordenador.

El BASIC MSX implementa un completo juego de instrucciones, que desde el AND hasta EQV, para manejar el Algebra de Boole en toda su extensión.

expresiones, donde no se comparan con ningún valor en concreto y cuyo resultado, sólo puede ser cierto (TRUE) o falso (FALSE). La sintaxis de la sentencia IF con este formato es:

IF (expresión) THEN (acción)

Este ejemplo se verá mejor con un ejemplo.

Muchas veces nos hemos visto en la situación de tener que comprobar si el resultado de una variable es 0 o no. Para ello tenemos esta instrucción, donde sustituyendo la expresión por la variable, obtendremos como resultado 0 ó 1, según la respuesta sea falsa o cierta respectivamente.

Por ejemplo, queremos determinar si la variable X de cualquier programa es 0, introduciremos la siguiente expresión:

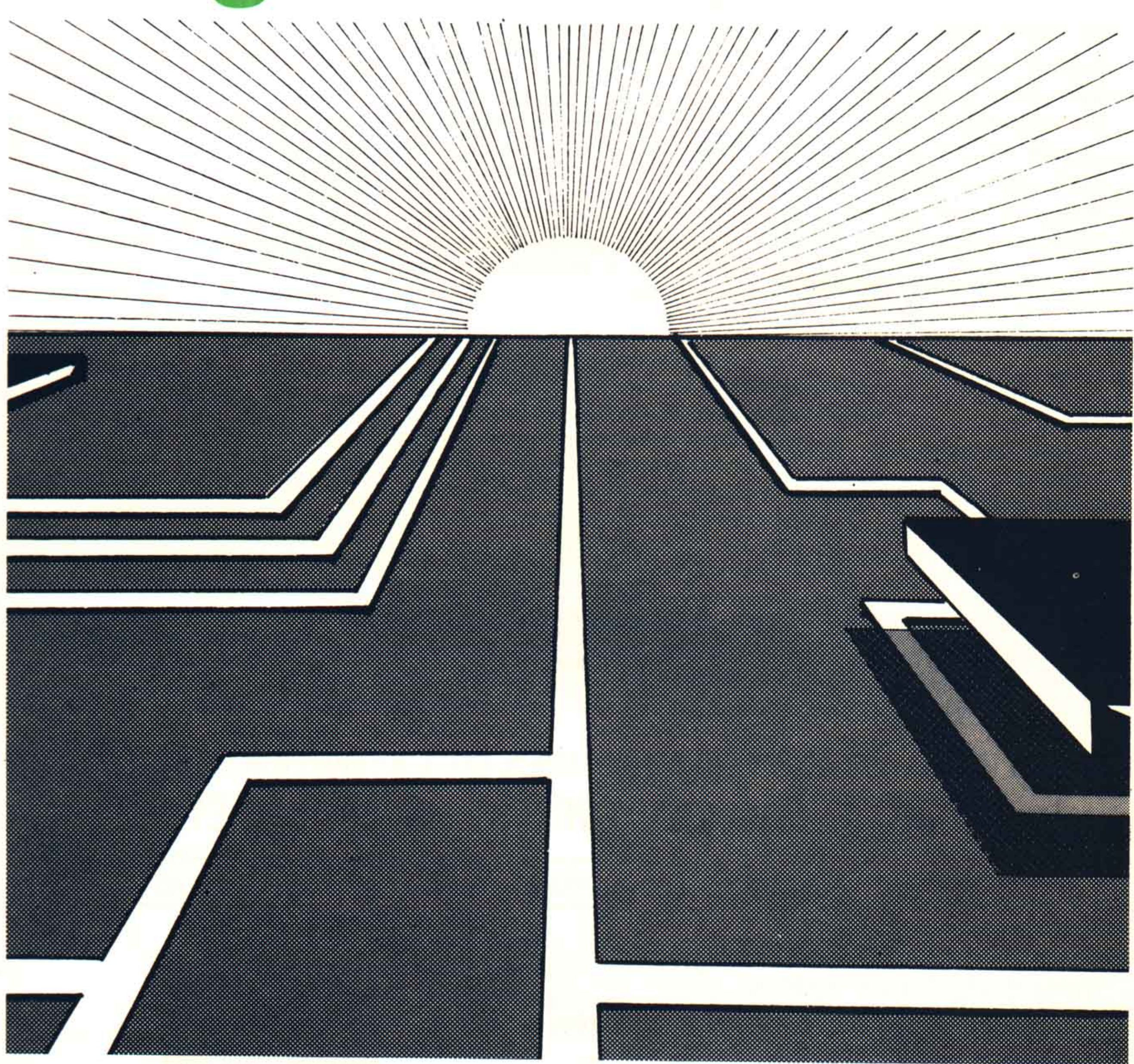
```
IF X THEN 2340
```

Con ella podremos saber si, efectivamente, la variable en cuestión vale 0 ó 1.

Con esta última explicación, finalizaremos la introducción dedicada al Algebra de Boole. Por supuesto que podríamos haber explicado más e incluso, haber profundizado más en la materia, pero esto nos llevaría a extendernos en demasia y no es cuestión dedicar tantas páginas a este tema. Nuestra misión ha sido la de describir otra herramienta más, como es el Algebra de Boole, para realizar programas junto con los comandos que la implementan. Ahora sólo queda, aunque suene repetitivo, que el lector, aplique estos conocimientos en la creación de sus programas.

PRINT 145 AND ((NOT 145) AND 58)	el	resultado	será	145
PRINT 145 OR (145 AND 58)		"	"	145
PRINT 58 AND 0		"	"	0
PRINT 59 AND 1		"	"	58
PRINT 145 AND ((NOT 145) AND 58)		"	"	0

Algebra de Boole



¿Quién no ha oído hablar del Algebra de Boole? Siendo una de la más importantes ramas de las matemáticas y sobre todo, herramienta ejemplar en el diseño de tablas de verdad y de circuitos lógicos, el Algebra de Boole, ha permitido a programadores y a diseñadores de circuitería, trabajar con más facilidad y soltura.

En programación, se usa para ayudar a entender la instrucción IF.

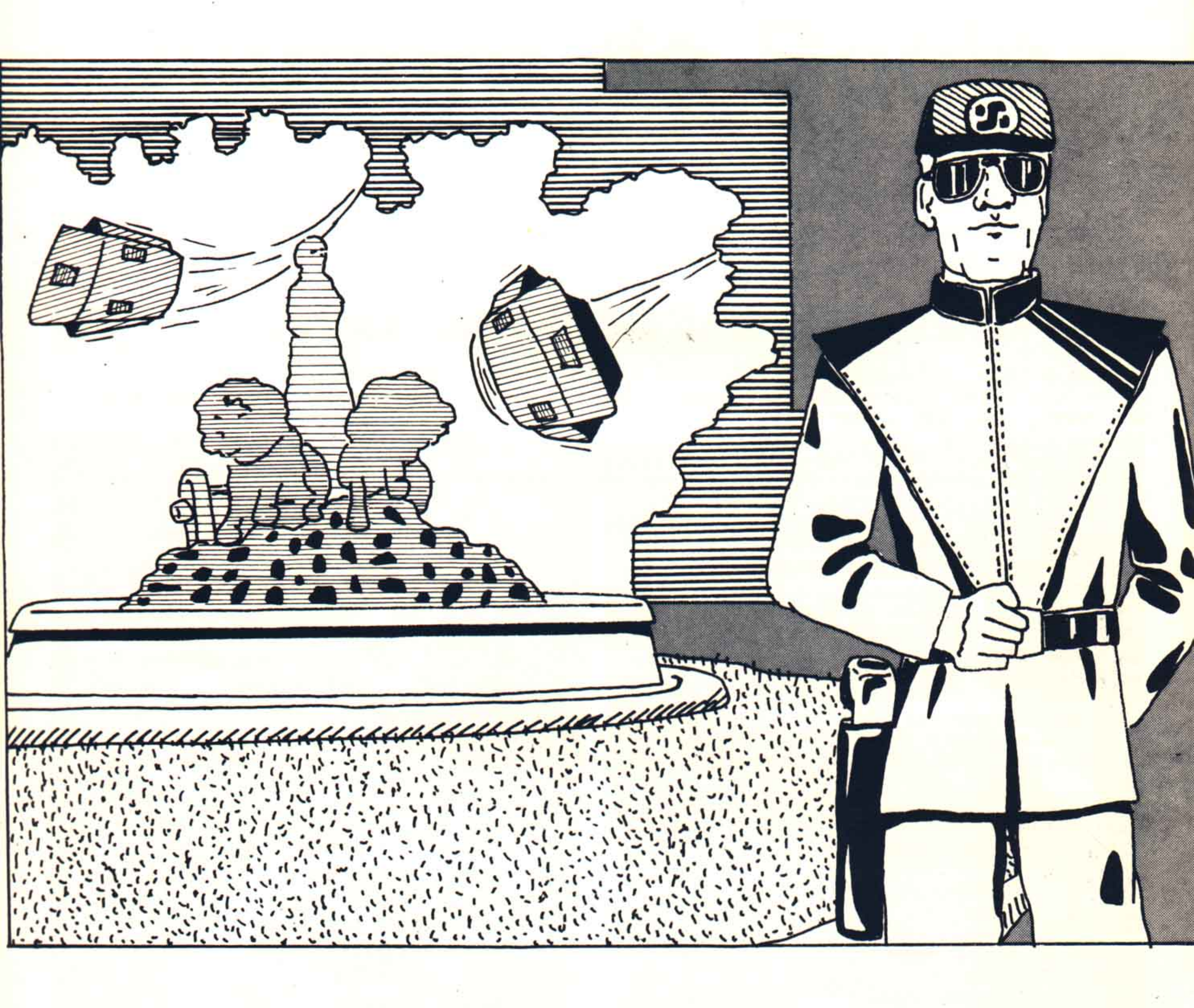
No es que esta sentencia sea difícil, sólo que cuando se complica mediante la incorporación de varios operadores lógicos, a veces nos las vemos y deseamos para comprenderlas.

Estos operadores, a los que debemos estar acostumbrados son: AND, OR, NOT, NAND y XOR, los cuales tienen su significado que son: Y, O, NO, Y negada y Or-exclusiva.

Una de estas tiene un desarrollo particular y único. A su vez, cada una de ellas tiene una tabla de verdad que le caracteriza.

Este programa mostrará todas las tablas posibles que se pueden realizar con esos operadores lógicos y con un total de cuatro variables.

```
5 CLS
10 PRINT "introduce el numero de variables a   utilizar ( hasta 4 )"
20 INPUT V
30 GOSUB 1000
40 PRINT: PRINT "introduce la funcion logica a reali- zar. Puedes elegir entre A
ND, NAND,  OR, NOR y XOR"
45 INPUT F$: CLS
46 FOR N=1 TO V: PRINT CHR$(N+64)+" ";:NEXT N
48 PRINT ,F$:PRINT "-----------------------------------"
50 IFLEFT$( F$,1)="n" THEN NEG=1
60 IF F$="and" OR F$="nand" THEN GOTO 2000
70 IF F$="or" OR F$="nor" THEN GOTO 3000
80 IF F$="xor" THEN GOTO 4000
90 PRINT "esa funcion no la conozco. Introduce otra que si conozca":GOTO 40
1000 REM contruccion de tabla de variables
1005 D=2^(V): DIM M$(D)
1010 FOR N%=1 TO D
1020 M$(N%)=BIN$(N%-1)
1030 NEXT
1040 REM normalizacion
1050 FOR N=1 TO D
1060 L=LEN (M$(N)):IF L=V THEN GOTO 1080 ELSE  FOR K=L+1 TO V:M$(N)="0"+M$(N):NE
XT K
1070 NEXT
1080 RETURN
2000 REM rutina and-nand
2010 X=2^V-1
2015 GOSUB 7000
2035 GOTO 6000
3000 REM funcion or-nor
3010 X=0: GOSUB 7000
3020 GOTO 6000
4000 REM funcion xor
4010 FOR N=1 TO D
4015 PR=VAL (("&b"+M$(N))):X=2^V-1
4020 S=1:NEG=0
4040 IF PR=0 OR PR=X THEN S=0
4050 GOSUB 5000
4055 NEXT N: GOTO 6000
5000 REM rutina de presentacion
5005 IF NEG=1 THEN  SWAP S,S1
5006 IF F$="or" OR F$="nor" THEN SWAP S,S1
5010 Z$=M$(N): FOR Q=1 TO V:PRINTMID$( Z$,Q,1)+" "; :NEXT Q: PRINT,S:RETURN
6000 LOCATE 0,21: PRINT "quieres otra (s/n)":INPUT R$:IF R$="s" THEN RUN ELSE CL
S:STOP
7000 FOR N=1 TO D
7010 S=0:S1=1
7020 IF X= VAL(("&b"+M$(N))) THEN S=1:S1=0
7030 GOSUB 5000
7040 NEXT N
7050 RETURN
```



Los visitantes en Madrid

Año; 2010. Lugar; España, concretamente Madrid. Los visitantes intentar reducir a las fuerzas rebeldes, pero la máxima concentración de fuerzas se encuentra cerca de Torrespaña. Estás al mando de un grupo de combatientes que tienen que eliminar el máximo número de extranjeros posibles.

Para ello, no hay más remedio que sustituir la antena parabólica que se utiliza en las transmisiones, por un cañon (este se encuentra dentro de la antena, para disimularlo de los ojos ajenos).

Hay varios tipos de naves, cada cual con su puntuación, consigue el recorde que aún nadie tiene. Para mover la antena, usa las teclas del cursor (izquierda/derecha) y dispara con la barra espaciadora.

No os hagaís ilusiones, ya que como buenos lagartos, estos se reproducen constantemente, por lo que el ataque será eterno. En este caso, la suerte no es necesaria, ya que aunque la tengas de cara, no servirá de mucho. Lo que sí necesitarás es buena puntería. ¡Qué la fuerza te acompañe!

```
10 REM LOS VISITANTES EN MADRID
20 REM ________________________
30 COLOR 1,0,4
40 SCREEN 1,2,0
50 CLEAR 1000
60 DEFSNG A
70 DEFINT B-R,T-Z
80 DEFSTR S
90 KEY OFF
100 SPRITE OFF
110 DIM SC(7),ST(8)
120 REM AH=100
130 GOSUB 2240
140 SPRITE$(10)=S1
150 SPRITE$(11)=S2
160 REM Instrucciones
170 FOR I=0 TO 9
180 PUTSPRITE I,(255,192),0,I
190 NEXT I
200 CLS
210 S="   LOS VISITANTES EN MADRID"
220 PRINT S
230 S="     ________________________"
240 PRINT:PRINT:PRINT
250 S="Las teclas del cursor giran"
260 PRINT S
270 S="la antena a la izq/dcha."
280 PRINT S
290 S="Dispara con el espacio."
300 PRINT S
310 PRINT:PRINT:PRINT
320 PUTSPRITE 10,(40,78),3,10
330 S="       50 Puntos"
340 PRINT S
350 PRINT:PRINT
360 PUTSPRITE 11,(40,102),5,11
370 PRINT S
380 PRINT:PRINT
390 PUTSPRITE 8,(40,126),9,8
400 S="       100 Puntos"
410 PRINT S
420 PRINT:PRINT:PRINT:PRINT
430 S="Pulsa una tecla para empezar"
440 PRINT S;
450 S=INPUT$(1)
460 PUTSPRITE 8,(40,208),0,8
470 PUTSPRITE 10,(40,208),0,10
480 CLS
490 REM Dibuja la ciudad
500 CLS
```

```
510 SOUND 7,&B10111000
520 LOCATE 0,24
530 FOR I=0 TO 7
540 PRINT SC(I)
550 NEXT I
560 FOR I=0 TO 6 STEP 6
570 FOR J=0 TO 1
580 LOCATE 6+I,2+J+I
590 FOR K=232 TO 238
600 PRINT CHR$(K+7*J);
610 NEXT K,J,I
620 LOCATE 0,0
630 PRINT "Rec.:";
640 PRINT USING"#######";AH
650 AP=0
660 CC=129
670 REM Creación de la estatua
680 SPRITE$(4)=ST(4)
690 XS=128
700 YS=143
710 REM
720 REM
730 PUT SPRITE 0,(124,159),15,1
740 PUT SPRITE1,(124,143),15,2
750 PUT SPRITE2,(125,168),15,0
760 REM NEXT
770 REM Creación de los sprites
780 CL=1
790 OS=-4
800 LOCATE 15,0
810 PRINT "Jug.:";
820 PRINT USING"#######";AP
830 S=SPRITE$(7)
840 IF S=S1 THEN S=S2 ELSE S=S1
850 SPRITE$(7)=S
860 XI=255
870 YI=16
880 CL=CL+2
890 IF CL>9 THEN CL=3
900 REM Movimiento del invasor
910 XI=XI+OS
920 I=OS
930 IF XI<-15 THEN OS=-OS
940 IF XI>255-OS THEN OS=-OS
950 IF OS<>I THEN YI=YI+8
960 PUTSPRITE 7,(XI,YI),CL,7
970 IF YI>182 THEN 2140
980 IF YI<112 THEN 1110
990 I=(YI AND 248)/8*32+32
1000 I=I+(XI AND 248)/8
```

```
1010 I=&H1800+I
1020 IF VPEEK(I)=32 THEN 1060
1030 VPOKE I,32
1040 CC=CC-1
1050 IF CC=0 THEN 2140
1060 I=I+3
1070 IF VPEEK(I)=32 THEN 1110
1080 VPOKE I,32
1090 CC=CC-1
1100 IF CC=0 THEN 2140
1110 IF B THEN 1190
1120 IF XI<4 OR XI>220 THEN 1360
1130 R=RND(-TIME)*4
1140 IF R THEN 1360
1150 REM Trayectoria del proyectil
1160 B=1
1170 XB=XI+8
1180 YB=YI+8
1190 YB=YB+6
1200 IF YB>191 THEN B=0
1210 PUTSPRITE 8,(XB,YB),CL,8
1220 I=(XB+5 AND 248)/8
1230 J=(YB+4 AND 248)/8*32+I
1240 J=&H1800+J
1250 K=VPEEK(J)
1260 IF K<224 OR K>227 THEN 1360
1270 VPOKE J,32
1280 PUTSPRITE 8,(XB,192),CL,8
1290 PUTSPRITE 9,(XB-3,YB-3),1,9
1300 GOSUB 2700
1310 YB=192
1320 CC=CC-1
1330 IF CC THEN 1200
1340 GOTO 2140
1350 REM Mueve el cursor
1360 FOR I=0 TO 2
1370 J=STICK(I)
1380 IF J THEN I=2
1390 S=INKEY$
1400 NEXT I
1410 T=0
1420 REM Mueve la antorcha
1430 XS=XS+(2 AND XS<136 AND J=3)
1440 XS=XS-(2 AND XS>120 AND J=7)
1450 SPRITE$(4)=ST((XS-120)/2)
1460 PUTSPRITE 4,(123,139),15,4
1470 YS=143+ABS(XS-128)
1480 IF F THEN 1640
1490 FOR I=0 TO 2
1500 T=STRIG(I)
```

```
1510 IF T THEN I=2
1520 S=INKEY$
1530 NEXT I
1540 IF T=0 THEN 1690
1550 SOUND 0,100
1560 SOUND 12,20
1570 SOUND 13,3
1580 H=XS-128
1590 V=-(8-ABS(H))
1600 F=1
1610 REM
1620 REM
1630 REM
1640 I=X>0 AND X<255 AND Y>YI
1650 IF I THEN 1710
1660 F=0
1670 H=0
1680 V=0
1690 X=XS-5
1700 Y=YS-13
1710 X=X+H
1720 Y=Y+V
1730 PUTSPRITE 6,(X,Y),15,6
1740 GOSUB 1800
1750 IF T=0 THEN 910
1760 SOUND 0,0
1770 GOSUB 2900
1780 GOTO 910
1790 REM Detecta la colisión
1800 IF F=0 THEN RETURN
1810 I=ABS(Y-YI)
1820 J=ABS(X-XI)
1830 K=1
1840 IF I<5 AND J<8 THEN 1890
1850 I=ABS(Y-YB)
1860 J=ABS(X-XB)
1870 IF I>8 OR J>4 THEN RETURN
1880 K=0
1890 PUTSPRITE 6,(XS,YS),15,6
1900 F=0
1910 H=0
1920 V=0
1930 X=XS
1940 Y=YS
1950 IF K THEN 2070
1960 PUTSPRITE 8,(XB,192),CL,8
1970 PUTSPRITE 9,(XB-3,YB-3),CL,9
1980 YB=192
1990 GOSUB 2700
2000 B=0
```

```
2010 AP=AP+100
2020 IF AP>=250000! THEN AP=INT(AP/100)
2030 LOCATE 22,0
2040 LOCATE 22,0:PRINT AP
2050 GOSUB 2780
2060 RETURN
2070 PUTSPRITE 7,(255,16),CL,7
2080 PUTSPRITE 9,(XI,YI-4),CL,9
2090 GOSUB 2700
2100 AP=AP+50
2110 GOSUB 2780
2120 RETURN 790
2130 REM Fin del juego
2140 BEEP
2150 FOR I=0 TO 10
2160 VDP(7)=15
2170 FOR J=0 TO 100
2180 NEXT J
2190 VDP (7)=4
2200 FOR J=0 TO 100
2210 NEXT J,I
2220 GOTO 170
2230 REM Define la ciudad y las nubes
2240 RESTORE
2250 FOR I=0 TO 31
2260 READ J
2270 VPOKE 1792+I,J
2280 NEXT I
2290 VPOKE 8220,177
2300 FOR I=0 TO 111
2310 READ J
2320 VPOKE 1856+I,J
2330 NEXT I
2340 VPOKE 8221,240
2350 VPOKE 8222,240
2360 FOR I=0 TO 7
2370 FOR J=0 TO 27
2380 READ K
2390 R=RND(-TIME)*4
2400 K=32 OR 224+R AND -K
2410 SC(I)=SC(I)+CHR$(K)
2420 NEXT J,I
2430 GOTO 2510
2440 REM Define los Sprites
2450 SP=""
2460 FOR J=0 TO 31
2470 READ S
2480 SP=SP+CHR$(VAL("&H"+S))
2490 NEXT J
2500 RETURN
```

```
2510 FOR I=0 TO 2
2520 GOSUB 2450
2530 SPRITE$(I)=SP
2540 NEXT I
2550 GOSUB 2450
2560 S1=SP
2570 GOSUB 2450
2580 S2=SP
2590 FOR I=6 TO 9
2600 IF I=7 THEN NEXT
2610 GOSUB 2450
2620 SPRITE$(I)=SP
2630 NEXT I
2640 FOR I=0 TO 8
2650 GOSUB 2450
2660 ST(I)=SP
2670 NEXT I
2680 GOTO 2780
2690 REM Explosión
2700 SOUND 0,0
2710 SOUND 3,0
2720 SOUND 7,&B10110000
2730 SOUND 12,50
2740 SOUND 13,3
2750 FOR I=0 TO 500
2760 NEXT I
2770 PUTSPRITE 9,(255,208),1,9
2780 SOUND 0,0
2790 SOUND 1,0
2800 SOUND 2,0
2810 SOUND 3,1
2820 SOUND 4,0
2830 SOUND 5,0
2840 SOUND 6,255
2850 SOUND 7,&B10111000
2860 SOUND 8,16
2870 SOUND 9,16
2880 SOUND 10,0
2890 SOUND 11,0
2900 SOUND 12,8
2910 SOUND 13,14
2920 RETURN
2930 REM Datos del programa
2940 DATA 0,96,96,0,0,0,0,0
2950 DATA 0,6,6,0,0,0,0,0
2960 DATA 0,0,0,0,0,96,96,0
2970 DATA 0,0,0,0,0,6,6,0
2980 DATA 0,0,0,0,0,0,3,15
2990 DATA 0,0,0,3,15,15,247,251
3000 DATA 0,0,62,255,255,255,255
3010 DATA 255,15,255,127,191,223
```

```
3020 DATA 223,223,188,224,254
3030 DATA 255,255,255,255,255,15
3040 DATA 0,0,0,128,240,63,255
3050 DATA 255,0,0,0,0,0,192
3060 DATA 224,31,63,63,63,31,15
3070 DATA 3,0,255,254,254,253
3080 DATA 251,247,129,0,255,255
3090 DATA 255,255,255,254,240,0
3100 DATA 251,247,215,185,127
3110 DATA 255,31,3,247,241,237
3120 DATA 251,255,255,255,248
3130 DATA 223,175,247,247,247
3140 DATA 227,128,0,240,248
3150 DATA 248,248,224,128,0,0
3160 DATA 0,0,0,0,0,0,0,0,0,0,1
3170 DATA 0,0,1,0,1,0,0,0,0,0,0
3180 DATA 0,0,0,0,0,0,0,0,0,0,0
3190 DATA 0,0,1,0,0,1,0,1,1,0,1
3200 DATA 0,0,1,0,1,0,0,0,0,0,0
3210 DATA 0,0,0,0,0,0,0,0,1,1,0
3220 DATA 1,0,1,1,1,1,0,0,1,0,1
3230 DATA 0,0,0,0,0,0,0,0,0,0,0
3240 DATA 0,1,0,1,1,1,1,0,1,1,1
3250 DATA 1,1,0,1,1,1,0,1,0,0,0
3260 DATA 0,0,0,0,0,1,0,1,1,1,1
3270 DATA 1,1,1,1,1,1,1,1,1,1,1
3280 DATA 1,1,1,0,1,0,0,0,0,0,0
3290 DATA 1,1,1,1,1,1,1,1,1,1,1
3300 DATA 1,1,1,1,1,1,1,1,1,1,1
3310 DATA 1,0,0,0,1,0,1,1,1,1,1
3320 DATA 1,1,1,1,1,1,1,1,1,1,1
3330 DATA 1,1,1,1,1,1,1,0,1,1,1
3340 DATA 1,1,1,1,1,1,1,1,1,1,1
3350 DATA 1,1,1,1,1,1,1,1,1,1,1
3360 DATA 1,1,1,1
3370 DATA F,F,F,F,F,F,F,F
3380 DATA F,F,F,F,F,F,F,F
3390 DATA E0,E0,E0,E0,E0,E0,E0,E0
3400 DATA E0,E0,E0,E0,E0,E0,E0,E0
3410 DATA 3,3,7,7,7,7,7,7
3420 DATA 7,7,7,7,7,7,7,7
3430 DATA e0,e0,f0,f0,f0,f0,f0,f0
3440 DATA f0,f0,f0,f0,f0,f0,f0,f0
3450 DATA 1,1,1,1,3,7,f,1f
3460 DATA 3f,3f,7f,7f,7f,7f,1f,7
3470 DATA 80,80,80,80,c0,e0,f0,f8
3480 DATA fc,fc,fe,fe,fe,fe,f8,e0
3490 DATA 3,7,1f,7f,b6,7f,1f,7
3500 DATA 3,0,0,0,0,0,0,0
3510 DATA 80,c0,f0,fc,da,fc,f0,c0
3520 DATA 80,0,0,0,0,0,0,0
3530 DATA 7,7,f,1f,35,7f,ff,f
```

```
3540 DATA f,0,0,0,0,0,0,0
3550 DATA c0,c0,e0,f0,58,fc,fe,e0
3560 DATA e0,0,0,0,0,0,0,0
3570 DATA 0,0,0,0,0,0,0,1
3580 DATA 1,0,0,0,0,0,0,0
3590 DATA 0,0,0,0,0,0,0,80
3600 DATA 80,0,0,0,0,0,0,0
3610 DATA 80,40,21,1e,1e,1e,1e,21
3620 DATA 40,80,0,0,0,0,0,0
3630 DATA 40,80,0,0,0,0,0,0
3640 DATA 80,40,0,0,0,0,0,0
3650 DATA 0,b,14,1d,2a,35,52,35
3660 DATA 34,4a,55,38,17,9,6,0
3670 DATA 80,e8,14,d2,2f,33,ed,a5
3680 DATA f7,47,8b,f3,95,2e,f0,0
3690 DATA 0,0,0,0,10,78,7f,7f
3700 DATA 78,10,0,0,0,0,0,0
3710 DATA 0,0,0,0,0,0,e0,e0
3720 DATA 0,0,0,0,0,0,0,0
3730 DATA 0,0,0,0,8,1c,3e,7f
3740 DATA 73,0,0,0,0,0,0,0
3750 DATA 0,0,0,0,0,0,0,0
3760 DATA c0,c0,0,0,0,0,0,0
3770 DATA 0,0,0,6,e,1e,1f,3
3780 DATA 1,0,0,0,0,0,0,0
3790 DATA 0,0,0,0,0,0,0,80
3800 DATA c0,c0,0,0,0,0,0,0
3810 DATA 0,0,f,7,f,3,3,1
3820 DATA 1,0,0,0,0,0,0,0
3830 DATA 0,0,0,80,c0,0,0,80
3840 DATA 80,c0,0,0,0,0,0,0
3850 DATA 3,3,7,1,1,1,1,1
3860 DATA 1,1,0,0,0,0,0,0
3870 DATA c0,c0,e0,80,80,80,80,80
3880 DATA 80,80,0,0,0,0,0,0
3890 DATA 0,0,0,1,3,0,0,1
3900 DATA 1,3,0,0,0,0,0,0
3910 DATA 0,0,f0,e0,f0,c0,c0,80
3920 DATA 80,0,0,0,0,0,0,0
3930 DATA 0,0,0,0,0,0,0,1
3940 DATA 3,3,0,0,0,0,0,0
3950 DATA 0,0,0,60,70,78,78,c0
3960 DATA 80,0,0,0,0,0,0,0
3970 DATA 0,0,0,0,0,0,0,0
3980 DATA 3,3,0,0,0,0,0,0
3990 DATA 0,0,0,0,10,38,7c,fe
4000 DATA ce,0,0,0,0,0,0,0
4010 DATA 0,0,0,0,0,0,7,7
4020 DATA 0,0,0,0,0,0,0,0
4030 DATA 0,0,0,0,8,1e,fe,fe
4040 DATA 1e,8,0,0,0,0,0,0
```



La pianola

Si Beethoven levantara la cabeza, seguro que diría barbaridades, algo así como que estamos estropeando la música, etc.

Pero a estas alturas, el lector tendrá algún que otro conocimiento sobre las posibilidades musicales del ordenador. De cualquier manera, si aún cree que ha visto poco, teclea este programa.

En el programa se dibuja un teclado de una pianola con buenas y diversas variantes musicales.

También incorpora un completo juego de instrucciones y totalmente auto explicativo, por lo que no hará falta comentar muy a fondo.

Lo mejor que puede hacer es practicar con los diversos tonos, teclas, duraciones, etc.

```
10 REM LA PIANOLA
20 CLS
30 COLOR 7,1,1
40 SCREEN3
50 OPEN"GRP:"AS 1
60 PRESET(65,60):PRINT#1,"PIANO"
70 KEY OFF
80 FOR I=1 TO 2000:NEXT I
90 SCREEN0
100 LOCATE12,10:PRINT"♫   P I A N O  ♪"
110 LOCATE25,0:PRINT"5896 Octetos"
120 PLAY"T255CDECDEEFGEFGGAGFECGAGFECCO3GO4C"
130 FOR I=1 TO 2000:COLOR,4:COLOR,2:NEXT
140 COLOR 7,1,1:BEEP
150 CLS:LOCATE0,10:INPUT"Deseas ver las instrucciones";R$
160 IF R$="s" OR R$="S"THEN 1170 ELSE 170
170 SCREEN2
180 REM * CLAVES *
190 LINE(23,120)-(256,192),15,BF
200 LINE(36,120)-(36,192),4
210 LINE(49,120)-(49,192),4
220 LINE(62,120)-(62,192),4
230 LINE(75,120)-(75,192),4
240 LINE(88,120)-(88,192),4
250 LINE(101,120)-(101,192),4
260 LINE(114,120)-(114,192),4
270 LINE(127,120)-(127,192),4
280 LINE(140,120)-(140,192),4
290 LINE(153,120)-(153,192),4
300 LINE(166,120)-(166,192),4
310 LINE(179,120)-(179,192),4
320 LINE(192,120)-(192,192),4
330 LINE(205,120)-(205,192),4
340 LINE(218,120)-(218,192),4
350 LINE(231,120)-(231,192),4
360 LINE(244,120)-(244,192),4
370 LINE(32,120)-(40,160),4,BF
380 LINE(45,120)-(53,160),4,BF
390 LINE(71,120)-(79,160),4,BF
400 LINE(84,120)-(92,160),4,BF
410 LINE(97,120)-(105,160),4,BF
420 LINE(188,120)-(196,160),4,BF
430 LINE(123,120)-(131,160),4,BF
440 LINE(136,120)-(144,160),4,BF
450 LINE(162,120)-(170,160),4,BF
460 LINE(175,120)-(183,160),4,BF
470 LINE(214,120)-(222,160),4,BF
480 LINE(227,120)-(235,160),4,BF
490 LINE(80,0)-(170,13),6,B
500 LINE(23,54)-(256,54),2
510 LINE(23,64)-(256,64),2
520 LINE(23,74)-(256,74),2
530 LINE(23,84)-(256,84),2
540 LINE(23,94)-(256,94),2
550 CIRCLE(33,84),2,7
560 PAINT(33,84),2,7
570 PRESET(90,4):PRINT#1,"P I A N O"
580 PRESET(105,25):PRINT#1,"M S X"
590 COLOR 1:PRESET(249,135):PRINT#1,"♪"
600 PRESET(249,160):PRINT#1,"♫"
610 COLOR 7:PRESET(67,0):PRINT#1,"♪"
620 PRESET(180,0):PRINT#1,"♫"
630 A$=INKEY$
640 REM * NOTAS *
650 COLOR 7
660 IF A$="A" THEN PLAY"C":PRESET(203,65):PRINT#1,"♪"
670 IF A$="S" THEN PLAY"D":PRESET(233,61):PRINT#1,"♪"
680 IF A$="D" THEN PLAY"E":PRESET(53,91):PRINT#1,"♪"
690 IF A$="F" THEN PLAY"F":PRESET(83,85):PRINT#1,"♪"
700 IF A$="G" THEN PLAY"G":PRESET(113,81):PRINT#1,"♪"
710 IF A$="H" THEN PLAY"A":PRESET(143,75):PRINT#1,"♪"
720 IF A$="J" THEN PLAY"B":PRESET(173,71):PRINT#1,"♪"
730 REM * OCTAVAS *
740  IF A$="1" THEN PLAY "O1"
750 IF A$="2" THEN PLAY "O2"
760 IF A$="3" THEN PLAY "O3"
```

```
770 IF A$="4" THEN PLAY "O4"
780 IF A$="5" THEN PLAY "O5"
790 IF A$="6" THEN PLAY "O6"
800 IF A$="7" THEN PLAY "O7"
810 IF A$="8" THEN PLAY "O8"
820 REM * SOSTENIDOS *
830 IF A$="Q" THEN PLAY "C#"
840 IF A$="W" THEN PLAY "D#"
850 IF A$="E" THEN PLAY "E#"
860 IF A$="R" THEN PLAY "F#"
870 IF A$="T" THEN PLAY "G#"
880 IF A$="Y" THEN PLAY "A#"
890 IF A$="U" THEN PLAY "B#"
900 REM * BEMOLES *
910 IF A$="I" THEN PLAY "C-"
920 IF A$="O" THEN PLAY "D-"
930 IF A$="P" THEN PLAY "E-"
940 IF A$="K" THEN PLAY "F-"
950 IF A$="L" THEN PLAY "G-"
960 IF A$="M" THEN PLAY "A-"
970 IF A$="," THEN PLAY "B-"
980 REM * LONGITUD *
990 IF A$="Z" THEN PLAY "T32"
1000 IF A$="X" THEN PLAY "T80"
1010 IF A$="C" THEN PLAY "T120"
1020 IF A$="V" THEN PLAY "T170"
1030 IF A$="B" THEN PLAY "T220"
1040 IF A$="N" THEN PLAY "T255"
1050 REM * MUSICA EN EL REGISTRO *
1060 IF A$="9" THEN PLAY"T250CDECDEEFGEFGGAGFECGAGFECCO3GO4C"
1070 IF A$="0" THEN A$="M2000S11T255O3CCEEGGAAA#A#AAGGEE":B$="FFAAO4CCDDD#D#DDCC
O3AA":C$="GGBBO4DDEEO3FFAAO4CCDD":PLAYA$+A$:PLAYB$+A$:PLAYC$+A$:PLAY"O4":BEEP
1080 IF A$="-" THEN PLAY"O4T255EDT120ET200DCO3BAT150G#T255AO4"
1090 GOTO 630
1100 REM * INICIALIZACION *
1110 IFA$=" " THEN PRESET(203,65):COLOR1:PRINT#1,"■":PRESET(233,61):PRINT#1,"■":
PRESET(53,91):PRINT#1,"■":PRESET(83,85);PRINT#1,"■":PRESET(113,81):PRINT#1,"■":P
RESET(143,75):PRINT#1,"■":PRESET(173,71):PRINT#1,"■"
1120 LINE(23,64)-(256,64),2
1130 LINE(23,74)-(256,74),2
1140 LINE(23,84)-(256,84),2
1150 LINE(23,94)-(256,94),2
1160 GOTO630
1170 REM * INSTRUCCIONES *
1180 CLS:BEEP
1190 LOCATE0,0:PRINT"Para tocar la escala, basta con pul- sar las teclas siguien
tes. A=DO, S=RED=MI, F=FA, G=SOL, H=LA, J=SI."
1200 LOCATE0,7:PRINT"Podrá tocar 8 octavas. 1=primera,    2=segunda, etc."
1210 LOCATE0,13:PRINT"Para alargar la nota, sitúese sobre  las letras Z, X, C, V
, B o N, siendo Z la más lenta y N la más rápida."
1220 LOCATE0,22:PRINT"Pulsa el espacio para continuar"
1230 S$=INKEY$
1240 IF S$=" " THEN 1260
1250 GOTO1230
1260 CLS:BEEP
1270 LOCATE0,0:PRINT"Podreis tocar los sostenidos, Q=DO#, W=RE#, E=MI#, R=FA#, T
=SOL#,  Y=LA#, U=SI#."
1280 LOCATE0,10:PRINT"Además, también se pueden tocar los  bemoles siguientes, I
=DOb, O=REb,     P=MIb, K=FAb, L=SOLb, M=LAb y ,=SIb"
1290 LOCATE0,14:PRINT"Así que si usted tiene que borrar     las notas de entrada,
 para poder vol-ver a componer, es suficiente con     pulsar el espacio."
1300 LOCATE0,22:PRINT"Pulsa el espacio para continuar"
1310 Z$=INKEY$
1320 IF Z$=" " THEN 1340
1330 GOTO 1310
1340 CLS:BEEP
1350 LOCATE0,0:PRINT"Como ejercicio tiene dos melodías    grabadas que tendrá qu
e generar. Es- tas se obtienen pulsando las teclas  9 y 0."
1360 LOCATE0,7:PRINT"¡Que lo paseis bien!"
1370 LOCATE0,22:PRINT"Pulsa el espacio para continuar"
1380 S$=INKEY$
1390 IF S$=" " THEN BEEP: GOTO 170
1400 GOTO 1380
1410 STOP:END
```



Continuando con las diversas rutinas que componen el complejo programa que está almacenado permanentemente en ROM, vamos a añadir algunas más a las que teníamos, esto no quiere decir que todo lo expuesto aquí sean rutinas en código máquina, pero por el momento, nada mejor para empezar, que ir comprobando el funcionamiento de estas rutinas.
Pero este mes, iniciaremos la sección con una pequeña idea enviada desde Málaga por nuestro seguidor Diego Féliz Gil.
Nos ha mandado un complemento a un dibujo generado en el artículo Diseño Gráfico, del número 1 de MSX Magazine. En él se explicó el funcionamiento de algunas instrucciones importantes y al final del artículo, se realizó un programa para mostrar dichos comandos.
Este dibujaba un pingüino, pero como el dibujo en sí quedaba demasiado frío, nuestro amigo nos ha enviado unas líneas de programa que añadidas al ya existente, dibujan un sombrero. De esta forma nuestro simpático dibujo queda algo más interesante.

Programa 1

Alguna vez nos hemos topado con el problema de no saber cuanto tiempo llevamos delante de la pantalla, con un juego o un programa. Si tenemos reloj a mano, tal problema no tendrá más consecuencia, pero si no es así, el siguiente programa resolverá la cuestión.
Este sencillo programa utiliza las interrupciones del sistema para visualizar un reloj en la esquina superior derecha de la pantalla, al mismo tiempo que ejecutamos nuestro programa.
Este se puede dividir en dos parte. La primera es la encargada de habilitar las interrupciones y pedir la introducción de la hora.
La segunda parte se situará al final del programa, poniéndo especial atención en no utilizar las mismas variables en el programas que en la rutina.

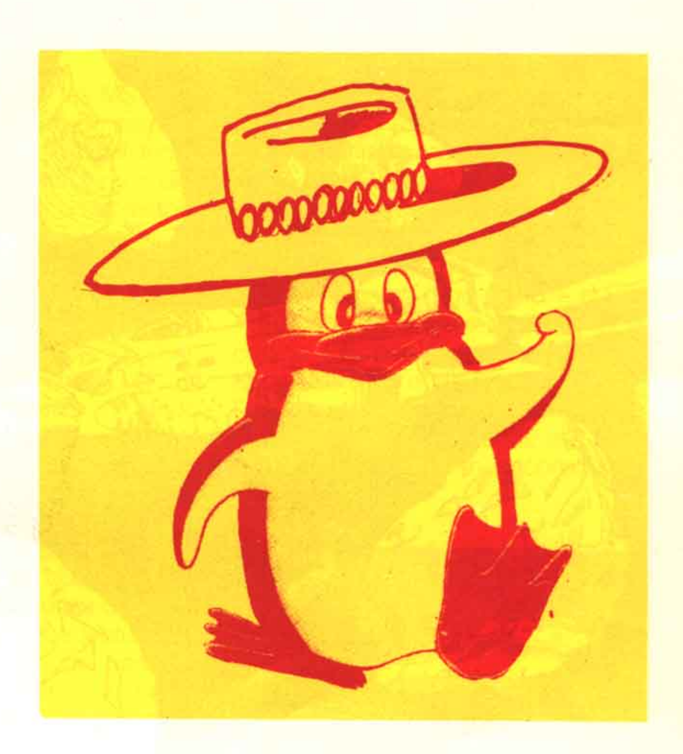
Programa 2

Ahora veremos unas cuantas rutinas más, a título de complemento a las expuestas el mes anterior, estas son equivalentes a lo que hacen dos instrucciones BASIC, que son, STRIG y STICK.
La primera de las rutinas es la utilizada por el intérprete BASIC situado en ROM, para limpiar la pantalla (CLS). Comienza en la direción C3 hexadecimal (195 decimal) y la podremos utilizar en nuestros programas, aplicando el CALL correspondiente.
Ahora veremos dos rutinas que son las

encargadas de leer el estado de los *port* de *joysticks*.
La primera comienza en la dirección D5 hexadecimal (213 decimal) y devuelve al acumulador el dato leído en el *port* correspondiente en ese registro antes de la llamada a la subrutina. En ensamblador se haría un programa de la siguiente manera:

```
LD A,1;joystick 1
CALL 213
CP1 JP Z, arriba
CP 3
JP Z, derecha
CP 5
JP Z, abajo
CP 7
JP Z, izquierda
RET
```

La que vamos a explicar a continuación, es el complemento de la anterior y sirve para comprobar si se ha pulsado algunos de los botones de disparo de cualquiera de los *joysticks* o la barra espaciadora. Comienza en la dirección D8 hexadecimal (216 decimal) y como la anterior, deberemos cargar en el acumulador el número de disparador que deseamos comprobar. A continuación, haremos el CALL correspodiente y la rutina nos devolverá en el acumulador 255, si se está pulsando el disparador y 0 en caso contrario.



```
10 REM RELOJ
20 INPUT "HORA";C%:INPUT"MINUTO";B%
30 CLS
40 ON INTERVAL=50GOSUB1000:INTERVALON
50 GOTO 50
1000 A%=A%+1:IF A%=60THENA%=0:B%=B%+1
1010 IFB%=60THEN B%=1:C%=C%=C%+1
1020 IFC%=13THEN C%=1
1030 LOCATE 22,0:PRINTC%;":";B%;":";A%
1040 RETURN
```

```
370 REM Rutina enviada por Diego Félix Gil
    de Málaga para la sección de Ideas y Trucos."
380 '
390 CIRCLE (100,35),17,3,,,.4
400 PAINT (100,30),3
410 CIRCLE (100,5),5,15,,,1.4
420 PAINT (100,5),15
430 LINE (83,35)-(117,35),9
440 LINE (83,35)-(100,10),9
450 LINE (100,10)-(117,35),9
460 PAINT (100,20),9
470 CIRCLE (100,35),17,3,,,.4
480 PAINT (100,30),3
490 GOTO 490
```



CODIGO

En capítulos anteriores aprendimos una pequeña parte del conjuto de instrucciones que nuestro ordenador es capaz de entender. Con ellas hicimos algunos sencillos programas que realizan sumas de diferents magnitudes. Por esto, evidentemente, es bantante poco para nuestras pretensiones. Además siempre que llamábamos a un programa en lenguaje máquina teníamos que hacerlo por medio de BASIC, pasándole los datos por medio de POKEs y sacando los resultados con PEEKs. Otro problema era el de no poder salvar los programas en cinta. Aunque esto no resulta grave de momento, ya que los que hemos dado solo son útiles a efectos didácticos, va a ser necesario cuando crezcan y se conviertan en algo útil, ya que queremos conservalos para no tener que introducirlos cada vez que queramos usarlos.

Un tercer problema (todavía no resuelto) se presenta por la complejidad que representa introducir un programa como códigos numéricos, debiendo hacer la traducción y conversión de todas las instrucciones y datos previamente. Sería mucho más fácil introducir las instrucciones 'LDA', 'ADD', etc, en lugar de sus correspondientes números. Esto en realidad se puede hacer por medio de unos programas especiales.

Hasta ahora hemos hablado indistintamente de lenguaje ensamblador y código máquina asumiendo que era lo mismo, pero hay grandes diferencias entre uno y otro. Diferencias que se explican ahora. Cuando nosotros hablabamos de instrucciones del tipos 'LDA 45000', estábamos usando el lenguaje ensamblador. En cambio cuando poníamos los códigos numéricos correspondientes (decimales o hexadecimales) estábamos usando código máquina.

Es decir, un programa ensamblador es aquel que nos permite usar las instrucciones por su nombre, escribir los números dieciseís *bits* como uno solo, sin tener que dividirlo en dos bloques de ocho *bits* y otra serie de ayudas que simplifican enormemente el trabajo. En cambio en código máquina todas estas ayudas no existen y tenemos que manejar los números. En esta serie lo que se ha hecho ha sido escribir el programa en ensamblador y traducirlo a mano, instrucción a instrucción, hasta obtener el código máquina.

Este proceso se puede automatizar mediante el uso de los denominados 'programas ensambladores' o más comúnmente 'ensambladores', qué son programas escritos con este fin específico de traducir de uno a otro. El único problema es que todavía no hemos encontrado ninguno para los ordenadores MSX (los otros ordenadores no le valen aunque lleven el mismo microprocesador), y por tanto tenemos que seguir haciéndolo como hasta ahora. Pero no desesperen, hemos enviado a toda la redacción (básicamente nuestro ex-

MAQUINA, ese desconocido (Cap. 4)

celso director) en busca de uno y esperemos que pronto nos encuentre alguno para anunciarlo en la revista y que todo el mundo lo pueda usar antes de enloquecer con los odiados códigos. Naturalmente como ustedes los lectores también son parte de la redacción (¿Lo dudaban?) esperamos su aviso si encuentran alguno antes que nosotros.

¿Qué es el estándar MSX?

Si usted se ha comprado un ordenador MSX, o piensa comprarlo, una de las razones que habrán influído será la de que es estándar y por tanto los programas de uno valen para otro. Esto implica que todos ellos tienen que tener unas características comunes a nivel interno y externo que los hagan compatibles. Pero pueden existir otros puntos en los que difieran (disposición interna de ciertos dispositivos, tipo de los dispositivos, etc.).

Muchas veces, como los programadores avanzados saben, es más fácil y rápido utilizar directamente algunos dispositivos que las rutinas previstas para ello, pero estos dispositivos, o su localización pueden no ser estándar y, por tanto, diferir de un modelo a otro. Para evitar hechos como éste, en esta serie recurriremos siempre a las disposiciones estándar tal como las ha definido **Microsoft** (la casa que diseñó el estándar y que se lo vendió posteriormente a los distintos fabricantes). Ya que todo ordenador con la etiqueta 'MSX' debe cumplir esas características. Esto a veces puede complicar un programa, pero no será mucho y siempre tendrá la ventaja de hacerlo compatible con todos los demás.

Por tanto, a partir de este mes iremos comentado las características estándar del MSX y como usarlas del mejor método.

Entradas y salidas

Una de las cosas que diferencian a un ordenador de otro, son las entradas y salidas. Mientras existen múltiples ordenadores con el mismo microprocesador (por ejemplo, el ZX Spectrum, el Amstrad y los MSX, llevan todos el Z-80) el tipo de pantalla que tienen, la disposición del teclado, como controlar el altavoz y la disposición de todos ellos en la memoria son distintas para cada marca de ordenador. En los MSX esto se ha estandarizado y todos llevan el mismo tipo de pantalla, el mismo tipo de teclado, el generador de sonido puede hacer lo mismo en todos, etc. Y lo que es más importante, el modo de manejarlos es el mismo en todos. Por lo que todo lo que digamos de ahora en adelante será válido para todos ellos.

Como dos primeros puntos a estudiar vamos a elegir los que nos faltaban para completar la rutina de suma de capítulos pasados. La lectura del teclado y la impresión de pantalla. Veremos cómo hacerlo y cómo transformar los datos para que sean útiles para nuestro programa.

El teclado, los caracteres ASCII

Ya vimos que en la memoria interna del ordenador sólo se podían almacenar números y, además, de 0 a 255. Entonces, ¿Cómo podemos guardar caracteres? La respuesta reulta sorprendentemente fácil, sobre todo si se ha jugado alguna vez a cifrar mensajes, ya que se basa en asignar un código numérico a cada letra o signo y almacenar ese código en la memoria. La única restricción que se le impone a este sistema de codificación es que no puede haber ningún número menor de cero o mayor que doscientos cincuenta y cinco, que es lo que cabe en un *byte*.

Con estas pautas se pueden generar infinidad de códigos distintos. La industria ha estandarizado unos cuantos de ellos. El EBCDIC usado principalmente en los *mainframes* (grandes ordenadores) de IBM, y el ASCII, que es el usan todos los microprocesadores y en especial los MSX, aunque en estos está modificada respecto al estándar. En la figura 1 se da la lista completa de todos los caracteres junto a su correspondiente código. Hay que tener en cuenta que desde el teclado algunos de estos caracteres se generan por medio de las teclas GRAF, CODE o mayúsculas. Además los códigos del 0 al 31 en el ASCII estándar cumplen funciones especiales de control de pantalla e impresora (por ejemplo, el RETURN que es el salto de línea, es el código 13), por lo que el ordenador debe distinguir entre un caso y otro. De momento sólo utilizaremos los realmente estándar (del 32 al 127) y posteriormente veremos como usar los demás.



Resulta muy importante tener en cuenta que con este sistema NO se almacenan números en la memoria (por lo menos con el significado que le vamos dando hasta ahora) sino códigos de caracteres que tienen distintos significado. Hay que observar que los símbolos de números (o, 1, 2, 3, 4, 5, 6, 7, 8 y 9) tiene asignados los códigos del 48 al 57 y se almacena cada uno en un *byte*. Esto nos indica que un número puede tener dos representaciones distintas, en binario como vimos al principio, en cuyo caso, cada *byte* puede almacenar un valor entre 0 y 255, y en ASCII, que sólo permite un carácter en cada *byte*. También se desprende de nuestra observación que el contenido de un *byte* puede ser un número (o parte de un número) expresado en binario o el código de un

PHILIPS MSX



El sistema más sabio

PHILIPS introduce en España el HOMECOMPUTER más sabio, el sistema MSX, nuevo estandard mundial.

¡Con cuanta sabiduría se ha pensado en cada una de sus características!

Con el PHILIPS MSX puede realizar mil combinaciones de elementos: monitores, impresoras, floppys, programas educativos, de juegos y aplicaciones profesionales, gracias a su compatibilidad total tanto en hardware como en software.

El PHILIPS MSX está tan sabiamente diseñado que Vd. puede elegir entre conectarlo al televisor de su casa, o a un monitor monocromo o de color.

De igual modo puede utilizar como unidad de almacenamiento de memoria un cassette normal o un Floppy Disc del sistema MSX.

¡Y qué potencia tiene el PHILIPS MSX!

Es tanta, que si lo utilizamos con un Floppy Disc y junto a MSX-DOS, es compatible con sistemas de tipo profesional y de precio mucho más elevado.

Y aquí no acaba la sabiduría con que ha sido creado el PHILIPS MSX.

Puede hacerlo crecer según sus necesidades, desde un sencillo ordenador doméstico, con el lenguaje Basic más potente del mercado, hasta un sistema de tipo profesional que puede llegar a una capacidad máxima de 1.024 K bytes.

PHILIPS MSX. Nunca se le quedará pequeño, nunca se le quedará anticuado.

PHILIPS MSX, creado como un equipo atractivo, fácil de usar y muy asequible de comprar.

¡PHILIPS MSX, sin duda, el sistema más sabio!

MSX-DOS es compatible con CP/M™ y posee la misma estructura de ficheros que MS-DOS™.

Todos los sistemas MSX son compatibles entre sí.

MSX, MSX-DOS™ y MS-DOS™ son marcas registradas de Microsof Corp.
CP/M™ es una marca registrada de Digital Research.

Si desea algún tipo de información relacionada con el campo del HOMECOMPUTER, estamos a su disposición en el teléfono

(91) 413 22 46

Desearía recibir más información sobre el PHILIPS MSX.

Nombre ....

Apellidos ....

Domicilio ....

....

PHILIPS IBERICA S.A.E.
Apartado de Correos 50.800
28080 MADRID

PHILIPS MSX HOMECOMPUTER SYSTEM

El amigo sabio de la familia.

CUSÍ

PHILIPS MSX HOMECOMPUTER SYSTEM

ESPECIFICACIONES TECNICAS

Consola VG 8010
Sistema MSX.
Teclado: Teclado con disposición y separación estilo profesional de 72 teclas.
Memoria: 32 K ROM, 48 K RAM (incluyendo 16 K RAM de vídeo).
Interconexiones incorporadas: Salida de RF, Salida Monitor, Interface audio-cassette, 2 conectores para controles manuales, 2 ranuras para cartuchos.

Consola VG 8020
Sistema MSX.
Teclado: De recorrido completo, profesional con 73 teclas.
Memoria: 32 K ROM, 80 K RAM (incluyendo 16 K RAM de vídeo).
Interconexiones incorporadas: Salida de RF, Salida Monitor, Interface audio-cassette, 2 conectores para controles manuales, 2 ranuras para cartuchos, Interface para impresora.

Características comunes VG 8010/VG 8020
Conjuntos de caracteres 253 alfanuméricos y gráficos (incluye la ñ).
Procesadores: Principal Z 80 A, Audio AY-3-8910, Vídeo TMS 9929 A.
Lenguaje BASIC MSX: 130 instrucciones incorporando macrocomandos y sprites.
Posibilidad máxima de expansión de memoria 1M. byte.
Editor de pantalla.
Utilizando MSX-DOS™ es compatible con CP/M™ y tiene la misma estructura de ficheros que MS-DOS™.

Monitor monocromo BM 7552 y BM 7502
Tubo de Imagen: Pantalla de alta resolución de 12", antideslumbrante, Fósforo P 42.
Ancho de Banda: 20 MHZ (a —3 dB).
Resolución: Horizontal: 920 líneas en el centro. *Vertical:* 285 pixels.
Caracteres en pantalla: 80×25 (2.000)
Salida Sonora: 0,3 W con 5% de distorsión.

Impresora de matriz
VW 0010, 40 columnas y VW 0020 de 80 columnas.
Método impresión: Matriz de puntos por impactos.
Matriz de carácter de 8×8 puntos.
Paso de caracteres 10,5 cpi y 10 cpi, respectivamente.
Velocidad de impresión 35 cps y 37 cps respectivamente.
Mecanismo PF alimentación por fricción y tracción.

Próximos lanzamientos
Monitor de color 14".
Floppy disc 3½" 500 K sin formatear (360 K formateado).

Software
Disponibles en MSX más de 150 títulos entre aplicaciones, utilidades, educativos y juegos en soporte ROM,, cassette y floppy de 3½".

carácter en ASCII. Evidentemente es necesario distinguir entre un caso u otro, ya que los significados son distintos en cada uno, pero por desgracia no existe ningún sistema que nos diga si el contenido de un *byte* cualquiera de la memoria es de uno u otro tipo.

Es más, además de estos dos casos existe un tercero, ya que ese *byte* puede ser una instrucción de un programa. El único modo de diferenciar entre todos los casos es estableciendo el programador el significado de cada uno y procurando que el programa no haga operaciones ilegales. Esto no representa una prohibición real de cara al ordenador y si el programa está mal hecho pueden producirse errores. Basta pensar en las rutinas de suma de capítulos anteriores. Si hubiésemos colocado erróneamente la dirección de almacenamiento de datos empezando en la memoria 40000, estaría en las mismas posiciones que el programa y al escribir encima nos lo borraría produciéndose un error (como se ve en la figura 2) al intentar ejecutar el microprocesador una instrucción que nosotros no le habíamos dicho. Por tanto hay que vigilar el contenido de cada memoria y recordar que clase de dato se almacena en ella.

Una máquina de escribir

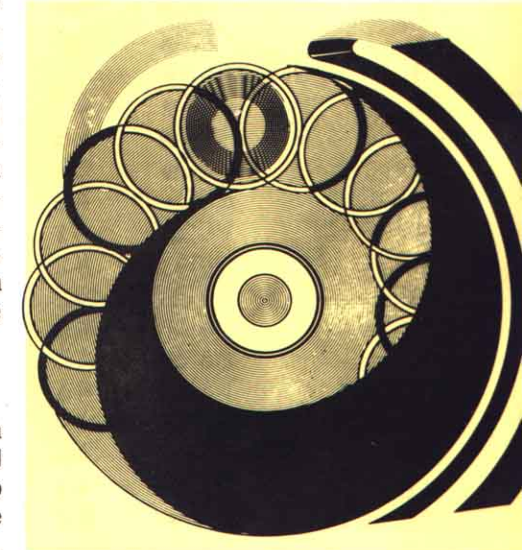
El primer programa que vamos a hacer es muy sencillo y su utilidad mínima, pero nos enseñará como manejar códigos ASCII. Básicamente consiste en una máquina de escribir, ya que lee los códigos de las teclas que pulsemos e imprime por pantalla el carácter correspodiente. Esto se sigue haciendo hasta que pulsemos un punto ".", en cuyo caso terminará la ejecución.

Para poder hacerlo vamos a aprender algunas instrucciones más, una de ellas compara el contenido del acumulador con un *byte* y pone las diversas banderas del registro 'F' de un modo u otro según resulte la comparación. Otra instrucción que vamos a aprender es la de llamada a subrutina. Esta nos permite llamar a otros programas en código máquina



para que ejecuten su acción y luego se devuelvan el control al terminar (siempre que tengan una instrucción RET al final). Estas rutinas pueden ser de las incluídas en la ROM de nuestra máquina o escritas por nosotros mismo y equivalen a los GOBUB...RETURN del BASIC que nos permite definir las operaciones más comunes aparte y luego llamarlas desde el programa. En ensamblador la llamada se realiza poniendo 'CALL' ('CDh o 205) seguido de la dirección de las subrutinas en dieciseís *bits*.

En este caso las rutinas que usaremos ya están definidas en la ROM y consisten en dos. La primera espera que pulsemos un carácter en el teclado y cuando lo hacemos devuelve



el control a la que le ha llamado con el código de dicho carácter en el acumulador. Se llama CHGET y se encuentra en la dirección '0009Fh'. La segunda imprime e la pantalla el carácter cuyo código se encuentre el en acumulador, se llama CHPUT y está en '00A2h'. Después de ver esto queda claro que el núcleo de nuestro programa consiste simplemente en llamar a la primera rutina y cuando nos devuelva el control llamar a la segunda, ya que el dato que nos interesa (el código del carácter) viene en el acumualdor de la primera y se tiene que enviar a la segunda

también en el acumulador sin ninguna modificación.

Después de esto se coge el carácter (que aún sigue en el acumulador ya esta segunda rutina no lo borra) y se compara con el código del punto (46 ó 2Eh') por medio de la instrucción 'CP', que compara el contenido del acumulador con el *byte* que se indique. Esta instrucción admite como muchas otras, varios tipos de direccionamiento. El que veremos ahora es el inmediato, que compara el *byte* que sigue a la instrucción. El formato de escritura en ensamblador puede ser de dos tipos. En el primero se pone la instrucción seguida del carácter entre comillas, en nuestro caso 'CP ".", el programa esamblador (o nosotros) pondrá el código de

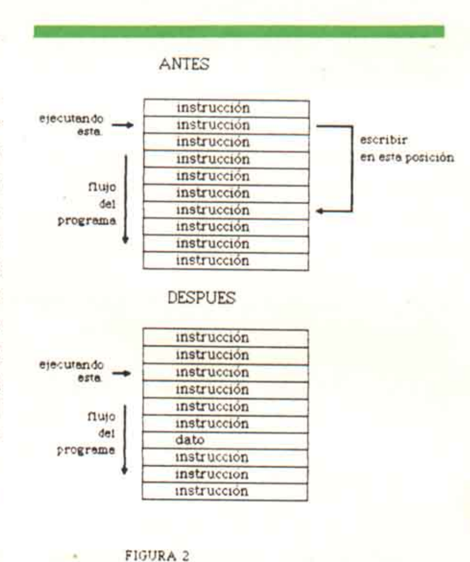


FIGURA 2

la instrucción seguido del código del carácter especificado. El segundo formato consiste en poner la instrucción y detrás del código del carácter, 'CP 46'. Este segundo formato tiene la ventaja de explicar claramente lo que se hace, pero para saber que carácter es tendremos que buscarlo en la tabla (Además deberemos saber que nos estamos refiriendo a un crácter y no a un número binario. En cualquiera de los dos casos el código generado será el mismo, por lo que el usuario queda libre de usar uno u otro sistema.

Una vez realiza la comparación, las banderas del registro F quedan como si se hubiese hecho una resta pero sin modificar el contenido del acumulador. La bandera 'N' siempre

0		33	!	66	B	99	c
1	☺	34	"	67	C	100	d
2	☻	35	#	68	D	101	e
3	♥	36	$	69	E	102	f
4	♦	37	%	70	F	103	g
5	♣	38	&	71	G	104	h
6	♠	39	'	72	H	105	i
7	•	40	(	73	I	106	j
8	◘	41	)	74	J	107	k
9	○	42	*	75	K	108	l
10	◙	43	+	76	L	109	m
11	♂	44	,	77	M	110	n
12	♀	45	-	78	N	111	o
13	♪	46	.	79	O	112	p
14	♫	47	/	80	P	113	q
15	☼	48	0	81	Q	114	r
16	+	49	1	82	R	115	s
17	┴	50	2	83	S	116	t
18	┬	51	3	84	T	117	u
19	┤	52	4	85	U	118	v
20	├	53	5	86	V	119	w
21	┼	54	6	87	W	120	x
22	│	55	7	88	X	121	y
23	─	56	8	89	Y	122	z
24	┌	57	9	90	Z	123	{
25	┐	58	:	91	[	124	¦
26	└	59	;	92	\	125	}
27	┘	60	<	93	]	126	~
28	X	61	=	94	^	127	
29	╱	62	>	95	_	128	Ç
30	╲	63	?	96	`	129	ü
31	+	64	@	97	a	130	é
32		65	A	98	b	131	â



puesta a uno, la 'C' se pone a uno si el operando externo es mayor que el acumulador y a cero en caso contrario. La 'P/V' es similar a la 'C' pero considerando números con signo. La 'H' lo hará de un modo similar co los cuatro *bits* de la derecha, la de signo 'S', como siempre, será igual al *bit* más significativo y la 'Z' se pondrá a uno solo si la resta da de resultado cero, es decir, si ambos son iguales. Esta última bandera es la que nos interesa en este caso, porque haremos uno u otra según sean iguales o no. Si son iguales hay que volver al principio para leer otro carácter. Esto se hace poniendo una instrucción 'JR NZ', que ya vimos en el capítulo pasado, a continuación dl 'CP' de modo que comprueba esta bandera 'Z' y si no está puesta se salta hacia donde se dice. Por último para acabar se pone un 'RET' después de ésta.

En la figura 3 se ve el listado completo en ensamblador (con los nom-

```
        ENSAMBLADOR                DECIMAL          HEXADECIMAL

INICIO  CALL    159                205 159 00       CD 9F 00
        CALL    162                205 162 00       CD A2 00
        CP      "."                254 46           FE 2E
        JR      NZ,INICIO          32 246           20 F6
        RET                        201              C9

FIGURA 3
```

132	ä	165	Ñ	198	▌	231	γ
133	à	166	ª	199	▞	232	Φ
134	å	167	º	200	▌	233	θ
135	ç	168	¿	201	▐	234	Ω
136	ê	169	⌐	202	▌	235	δ
137	ë	170	¬	203	//	236	∞
138	è	171	½	204	\\	237	φ
139	ï	172	¼	205	▼	238	ε
140	î	173	¡	206	▲	239	∩
141	ì	174	«	207	▶	240	≡
142	Ä	175	»	208	◀	241	±
143	Å	176	Ã	209	⧗	242	≥
144	É	177	ã	210	⋈	243	≤
145	æ	178	Ĩ	211	▪	244	⌠
146	Æ	179	ĩ	212	▪	245	⌡
147	ô	180	Õ	213	▪	246	÷
148	ö	181	õ	214	▪	247	≈
149	ò	182	Ũ	215	✱	248	°
150	û	183	ũ	216	Δ	249	∙
151	ù	184	Ĳ	217	ǂ	250	·
152	ÿ	185	ĳ	218	ω	251	√
153	Ö	186	¾	219	█	252	ⁿ
154	Ü	187	∽	220	▄	253	²
155	¢	188	◊	221	▌	254	■
156	£	189	‰	222	▐	255	
157	¥	190	¶	223	▀		
158	₧	191	§	224	α		
159	ƒ	192	▁	225	β		
160	á	193	▚	226	Γ		
161	í	194	▆	227	π		
162	ó	195	▔	228	Σ		
163	ú	196	▪	229	σ		
164	ñ	197	■	230	µ		

bres de las instrucciones) y en código máquina (decimal y hexadecimal). Observando el listado en ensamblador se puede ver que en el salto no se pone el desplazamiento como hacíamos el mes pasado, sino que se indica un nombre y ese mismo nombre está delante de la instrucción a la que queremos ir. Esta palabra se denomina etiqueta y simplifica mucho el trabajo cuando se está trabajando con un programa ensamblador ya que éste se encarga automáticamente de calcular el desplazamiento. En cualquier caso aclara el listado para leerlo. Como norma siempre utilizaremos las etiquetas en el listado en ensamblador y el desplazamiento en el listado en código máquina, de este modo el programa será fácilmente comprensible y los códigos se pueden introducir inmediatamente.

El programa se puede introducir como es habitual en la dirección 4000 por medio del cargador hexadecimal. Para ejecuarlo se utilizan las habituales instrucciones:

```
DEFUSRO = 4000
PRINTI USRO (1)
```

A continuación cualquier carácter que tecleemos aparecerá en la pantalla como si estuviésemos trabajando con una máquina de escribir. Al pulsar un punto se terminará de ejecutar la rutina y después de imprimir 'OK' el ordenador volverá a su modo de funcionamiento normal.

Como salvar nuestros programas

Vamos a explicar ahora como salvar y leer programas en código máquina en cinta. Esto resulta fundamental cuando se escriben programas largos e interesantes para evitar tener que introducirlos de nuevo. El BASIC del MSX tiene dos instrucciones especiales para esto. La primera es:

BSAVE "nombre del programa", dirección inicial, dir. final, dir. ejec.

La dirección inicial es la menor ocupada por el programa (en los que hemos hechos, ésta es donde empieza a ejecutarse). La dirección final es la más alta ocupada por el programa. El modo de hallarla es muy sencillo. Se cuenta el número de *bytes* que ocupa el programa y se le suma a la dirección inicial, al total se le resta uno y ésta es la dirección final. Por último la dirección de ejecución es donde se empieza a ejecutar el programa cuando lo cargamos. Si no se pone esta dirección se considera que es la misma que la dirección inicial.

Así, por ejemplo, para salvar el programa que acabamos de crear con el nombre 'ECO', habría que escribir:

```
BSAVE "ECO", 40000, 40010
```

La dirección de ejecución no se poe ya coincide con la dirección inicial.

Para cargar un programa previamente grabado se utiliza:

BLOAD "nombre del programa", R desplazamiento

El nombre tiene que ser el mismo con el que se salvó, si se pone la R el programa se empezará a ejecutar automáticamente al terminar de cargarse. El desplazamiento es opcional y si no se pone el programa se cargara la dirección que se salvó, pero si se pone, esa dirección se suma a la que se usa para salvarlo y el programa se carga a partir de ésa. No conviene usar esa última opción ya que muchas veces el programas utiliza direcciones absolutas en los saltos y ya no serián válidos al modificar su

ENSAMBLADOR			BASIC
INICIO	CALL	159	10 A$=INKEY$
	CP	"a"	20 IF A$<"a" THEN GOTO 50
	JR	C,IMPRI	
	CP	"{"	30 IF A$>="{" THEN GOTO 50
	JR	NC,IMPRI	
	SUB	#32	40 A$=CHR$(ASC(A$)-32)
IMPRI	CALL	162	50 PRINT A$;
	CP	"."	60 IF A$="." THEN GOTO 10
	JR	NZ,INICIO	
	RET		70 END

FIGURA 5

ENSAMBLADOR			DECIMAL	HEXADECIMAL
INICIO	CALL	159	205 159 00	CD 9F 00
	CP	"a"	254 97	FE 61
	JR	C,IMPRI	56 6	38 06
	CP	"{"	254 123	FE 7B
	JR	NC,IMPRI	48 2	30 02
	SUB	#32	214 32	D6 20
IMPRI	CALL	162	205 162 00	CD A2 00
	CP	"."	254 46	FE 2E
	JR	NZ,INICIO	32 236	20 EC
	RET		201	C9

FIGURA 6



posición en memoria. O también puede ser que se ponga encima del área de datos, creado problemas como vimos anteriormente.

Para cargar y ejecutar el programa anterior sería:

```
BLOAD "ECO", R
```

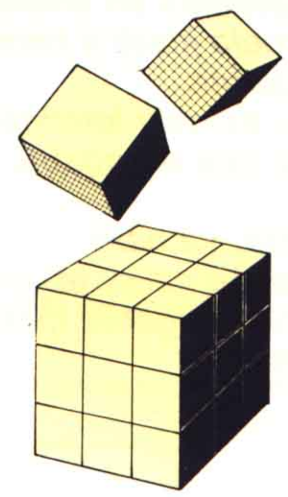
Minúsculas y mayúsculas

El siguiente programa que vamos a hacer nos permtira profundizar un poco más en el tema de las comparaciones así como en el manejo de los caracteres ASCII, especialmente veremos como pueden ser tratados como tales o como números cuando nos interesa.

Esta rutina realizará la misma operación de lectura y escritura de antes pero además convertirá todas las minúsculas a mayúsculas antes de imprimirlas. El primer paso a realizar será, como antes, obtener el carácter del teclado, a continuación mirará si es un miyúscula y si lo es la convertirá en mayúsculas, en caso de que no lo sea (que ya sea mayúscula, o cualquier otro símbolo) lo imprimirá directamente.

La operación de leer un carácter del teclado se realizará como antes por medio de la rutina CHGET, a continuación se realizan dos compartaciones, la primera con 'a' y la segunda con ' ' (el carácter posterior a la 'z') . En el primer caso, si la bandera de acarreo 'C' está a cero indica que el carácter del acumulador es mayor o igual a la 'a', en le segundo caso la bandera de acarreo a uno señala que este carácter es inferior a ' '. Si se cumplen las dos comparaciones entonces es una letra minúscula. En el programa lo que se hace es realizar la primera comparación , a continuación se realiza un salto condicional a la parte de impresión si la bandera de acarreo está a uno y si no se realiza la segunda comparación, a continuación de la cual se realiza otro salto condicional a la posición de impresión si la bandera de acarreo está a cero. En caso de que no se haga el salto se realiza la conversión a mayúsculas.

Este proceso de conversión es muy sencillo si se tiene en cuenta que los códigos de las mayúsculas tienen la misma ordenación y que los caracteres respectivos se diferencian en 32 (la 'A' tiene el código 65 y la 'a' el 97, que es 65+32, la 'B' tiene el 66 y la 'b' el 98, etc). Ya que sabiendo que es una minúscula, basta restarle 32 para obtener su correspondiente mayúsula. En este caso tratamosal *byte* como si fuese un número binario pero se hace conscientemente de que lo que manejamos es un código ASCII y la operación se



realiza para obtener otro código ASCII.

Por último y una vez convertida a mayúscula, se llega a la parte de impresión que es donde se va desde los dos saltos incondicionales anteriores. Aquí se vuelve a recurrir a la rutina CHPUT que vimo antes. Después de esto se comprueba, como antes, si es el carácter de terminación (el punto, aunque puede ser cualquier otro cambiando ese dato) y volviendo al principio si no lo es.

En este programa se utilizan algunas instrucciones nuevas. Las dos primeras son 'JR C' y 'JR NC', que realizan saltos incondicionales del mismo modo que el 'JR NZ' que ya hemos usado, pero comprobando la bandera de acarreo y saltado si está a uno u otro cero respectivamente. Los códigos de operación correspodientes so '38h' (56) para 'JR C' y '30h' (48) para 'JR NC'. Ambas deben ir seguidas del desplazamiento que sumado al contador del programa nos de la siguiente dirección de ejecución.

La otra instrucción nueva, que vamos a usar es la de restar. Ésta tiene un formato y manejo similar a las de sumar pero restando el número indicado del que está en el acumulador. Al igual que con la suma puede haber resta con o sin *bit* de acarreo, pero ahora consideraremos sólo la que no lo usa. El nombre de la instrucción es 'SUB' y el código cuando se realiza con direccionamiento inmediato es 'D6h' (214) que debe ir seguido del *byte* a restar.

En la figura 5 se da el listado en ensamblador del programa y al lado se ha metido un 'pseudo listado' en BASIC en el que se pueden ver las instrucciones equivalentes. Aunque este listado funciona si se introduce en el ordenador, su finalidad es ver, paso a paso, lo que hace la rutina comparándola con un lenguaje más conocido como es el BASIC.

En la figura 6 se vuelve a dar el listado en ensamblador pero acompañado esta vez por los códigos decimales y hexadecimales.

Resumen del capítulo

Este mes hemos visto como se pueden realizar comparaciones y saltos condicinales en base a estas comparaciones. Como llamar a subrutinas y como restar dos *bytes*. Así mismo se ha visto el código ASCII y su funcionamiento por medio de dos programas que hacían eco del teclado a la pantalla. Aquellos lectores que crean interesante convertir su ordenador en ua máquina de escribir de verdad y posean impresora, pueden modificar rápidamente el primer programa sustituyendo la dirección de salto de impresión por '00A5h', que es la dirección para sacar el carácter por impresora, en lugar de por pantalla. También les convendría el carácter de determinación (el punto) por algún otro, esto lo dejamos a su libre elección.

Fernando García

Busy

A estas alturas, ya estareís familiarizados con el singular personaje que aparece en los monentos claves (y no tan clave) de la revista. Se trata de nuestra mascota BUSY, de todos modos, será él quién se presente.
Nos acompañará a lo largo de todos los números de MSX, donde le veremos en artículos, programas, etc, siempre se cuela justo a tiempo para vosotros le tengaís en cuenta.
Nuestro simpático amigo, que es muy inteligente, procurará hacer que la revista sea agradable e interesante de ver.
Además, como es muy coqueto, ya está pensando en regalar algo a quién mejor le dibuje. Claro que, como es muy exigente quiere que se realice con un programa.

¡HOLA!. LOS CHICOS DE LA MSX ME HAN DICHO QUE ME VISTA DE ETIQUETA Y ME PRESENTE YO MISMO. PUES BIEN, ME LLAMO BUSY, (NO LO OLVIDEIS) TENGO DOS AÑOS, SOY SOLTERO, CINCO DIOPTRIAS EN CADA OJO, Y UNAS ANTENAS DE DELICADA TEXTURA. EL PISO DONDE VIVO ES UN POCO PEQUEÑO: 2 cm² habitables.
TE SOBRA.
NO INTERRUMPAS, GUAPO. ME ENCANTA LA INFORMATICA Y SOBRE TODO PROGRAMAR CON LOS MSX. ADEMAS TENGO IMPRESORA, UNIDAD DE DISCOS, INTERFACES, SOFTWARE CANTIDAD...
DOS CUERNOS....
NO INTERRUMPAS GUAPETE. BUENO YA SABEIS DONDE ESTOY Y..:
A MANDAR QUE PARA ESO ESTAMOS
....AQUÍ EL GUAPIN ME VA A HACER PERDER LA COMPOSTURA

Rincón del lector

INHIBIR LA TECLA DE STOP

Si bien el BASIC de Microsoft, contempla la posibilidad de inhibir las interrupciones durante la ejecución de los programas en BASIC, mediante la bifurcación ON STOP para la protección de programas, no hay modo de conseguir inhibir la tecla STOP, que si bien es útil en ciertos casos, no deja de ser molesta, a mi entender, en la mayoría de ellos. Seguramente, para su anulación, deberá recurrir a algún tipo de rutina en código máquina o bien cambiar algunos valores de la RAM del sistema, pero ante la posibilidad de conocer esta última, les agradecería me solucionasen el problema.

Por último, les gradecería, me informasen también de si existe algún modo de almacenar programas en cinta con auto ejecución.

Xavier Casajuana Mogas
Granollers

Si desea desabilitar la tecla STOP solamente, no le quedará más remedio que modificar la rutina de lectura del teclado del sistema, tarea imposible de realizar, a menos de que se tenga un buen conocimiento del funcionamiento de este.

Respecto a la autoejecución de los programas en *cassette*, la única forma que conocemos es grabándo el programa en la instrucción:

– SAVE "CAS: nombre "

y cargarlo mediante la instrucción:

– LOAD "CAS: nombre ", r

La r al final indica al ordenador que ejecute (run) el programa en cuanto finalice su carga.

LISTADOS ERRONEOS

La presente es para comunicarles mi profunda decepción cuando al introducir los programas del primer número de su re-

vista en mi ordenador, éste me respondía con mensajes de error, en casi todas las líneas.

Me gustaría que me explicara la causa de estos errores que considero muy graves en una revista con sus pretensiones.

Julio Antonio Pérez Morilla
Granada

Es difícil comprender que ningún listado de los publicados en la revista, haya funcionado correctamente. Además en tu carta, no especificas ni el programa que te falló, ni la línea en que se cometió el presumible error.

Los programas publicados se han probado en los ordenadores mencionados en la anterior respuesta, por lo que nos parece difícil entender semejantes fallos. Todos se han comprobado, más de una vez y por más de una persona.

MSX CON EL CHIP Z800

He leído en vuestra revista que se esperan los nuevos modelos MSX con un procesador Z800 de 16 *bits* y un *chip* de gráficos tipo 9229. Me gustaría saber si van a tardar en salir y cuando aparecerán en el mercado español.

Fidel Recio González
Madrid

La segunda generación de ordenadores MSX ya se ha presentado en Japón. Las mejoras son, como ya apuntas, mayor potencia a todos los niveles, pero sobretodo un *chip* de video que permite controlar 128K de video. Por el momento y debido al retraso exsitente en la creación del chip Z800, parece ser que la generación de ordenadores MSX con procesadores de 16 *bits*, todavía anda algo lejos. Desde luego, la idea de los japoneses es realizar versiones



más potentes de este estándar, lo que redundará en beneficio de todos los usuarios de estos ordenadores o de los que van a comprar uno, ya que la compatibilidad se va a mantener.

De aparecer en el mercado español, esto sería imprescindible, ya que igual al año que viene tenemos los primeros ordenadores de la segunda generación, como a finales de año. No existe fecha concreta, pero hay que tener en cuenta que en el mercado español lleva un atraso de bastantes años con respecto de los avances tecnológicos de otros países y más si se trata del Japón.

Si ellos ya están en la segunda generación, nosotros todavía estamos empezando a recorrer el camino, aunque tenemos la particularidad de hacerlo a pasos de gigante.

En suma, la aparición de estos nuevos ordenadores todavía está sin aclarar.

DIRECTOR:
Juan Arencibia
COORDINADOR EDITORIAL:
Emiliano Juárez
REDACCION:
Fernando García, Santiago Gala, Ricardo García, Teresa Aranda, Francisco Mancera.
DISEÑO:
Ricardo Segura y Benito Gil.
Editada por
PUBLINFORMATICA S.A.
PRESIDENTE:
Fernando Bolín.
DIRECTOR EDITORIAL:
Norberto Gallego.
Administración:
PUBLINFORMATICA
GERENTE DE CIRCULACION Y VENTAS:
Luis Carrero.
PRODUCCION:
Miguel Onieva.
DIRECTOR DE MARKETING:
Antonio Gonzalez.
SERVICIO AL CLIENTE:
Julia Gonzalez.
Tel. 733 79 69
ADMINISTRACION:
Miguel Atance,
JEFE DE PUBLICIDAD:
María José Martín.
DIRECCION Y REDACCION:
C/ Bravo Murillo, 377 - 5° A
Tel. 733 74 13
28020 MADRID
PUBLICIDAD Y ADMINISTRACCION:
C/ Bravo Murillo 377 - 3° E
Tel. 733 96 62-96
Publicidad en Madrid:
Fernando Hernando
Publicidad en Barcelona:
Olga Martorell
C/ Pelayo, 12
Tel. (93) 301 47 00 Ext. 27-28.
08001 BARCELONA
Deposito Legal: M. 16.755-1985
Impreso en Héroes S.A.
C/ Torrelara, 8. 28016-MADRID.
Distribuye:
S.G.E.L. Avda. Valdelaparra s/n.
Alcobendas (Madrid).

SUSCRIPCIONES:
Rogamos dirija toda la correspondencia relacionada con suscripciones a:
MSX
EDISA: Tel. 415 97 12
C/López de Hoyos, 141-5.°
28002 MADRID
(Para todos los pagos reseñar solamente MSX)
Para la compra de ejemplares atrasados diríjanse a la propia editorial
MSX
C/Bravo Murillo, 377-5.° A
Tel. 733 74 13 28020 MADRID

Si deseas colaborar en MSX remite tus artículos o programas a Bravo Murillo 377, 5.° A. 28020 Madrid. Los programas deberán estar grabados en cassette y los artículos mecanografiados.
A efectos de remuneración, se analiza cada colaboración aisladamente, estudiando su complejidad y calidad.

Programas Sony para ordenadores MSX

A la orden.


Monkey Academy


Países del Mundo-1


Países del Mundo-2


Computador Adivino


Computer Billiards


The Snowman


Cubit


Character Collection


Stop the express (Para el Tren)


Hustler (Billar Americano)


Data cartridge


Quinielas y Reducciones


Home Writer


Sparkie


Aprendiendo Inglés-1


Binary Land


Creative Greetings


Aprendiendo Inglés-2


Antartic Adventure


Mastermind


Contabilidad Personal


Athletic Land


E.I.


Ficheros


El Ahorcado


Dorodon


La Pulga


Cosmos


Control de Stocks


Battle Cross


Mouser


Crazy Train


Ali baba


Juno First


Car Jamboree


Tutor


Track and Field-1 (olimpiadas)


Blackjack


Track and Field-2 (olimpiadas)


Driller Tanks (Tanque Destructor)


Sonygraph


Ninja (El Samurai)



Les Flics

Y muchos más títulos

Ordenador Doméstico

HIT BIT

SONY

Para lo que guste ordenar.



MSX



GoldStar MSX

MEMORIA RAM DE USUARIO: Una potente memoria de 64K le dará la fuerza necesaria para ejecutar los mejores programas del mercado.

CONECTORES DE EXPANSION: Aseguran la conexión a gran cantidad de periféricos como impresoras, diskettes y joysticks.

ROM y VIDEO ROM: Permiten al Goldstar ejecutar y trabajar con potentes programas de gráficos sin tener que utilizar la memoria RAM.

En el PORT DE CARTUCHOS podrá conectar todos los programas MSX existentes, simplemente introduciendo el cartucho —¡olvídese de esas complicadas cintas!

La FUENTE DE ALIMENTACION está incorporada al ordenador, de manera que no tendrá que manejar ni ocultar transformador alguno.

EL TECLADO es del tipo QWERTY, con la incorporación de teclas de función y del control del cursor.

EL SONIDO es una de las mejores características del Goldstar —con 5 octavas y un sin fin de tonos increíbles.

¡¡49.500 pts.!!



PAMPLONA:
C/Alfonso el Batallador, 16 (trasera)
Tel. 27 64 04 C. Postal 3107

SAN SEBASTIAN:
Plaza de Bilbao, 1.
Tel. 42 62 37 - Télex 38095-IAR
C. Postal 20005